Departamento de Lenguajes y Ciencias de la Computación Universidad de Málaga

Administración de Bases de Datos

(Ingeniería Técnica en Informática de Gestión)

Conceptos sobre Administración del SGBD Oracle

E.T.S.I. Informática ***J. Galindo Gómez***

El SGBD ORACLE

Bibliografía: La mejor sobre este tema es, aunque está en inglés, el **Manual de Oracle** de la última versión.

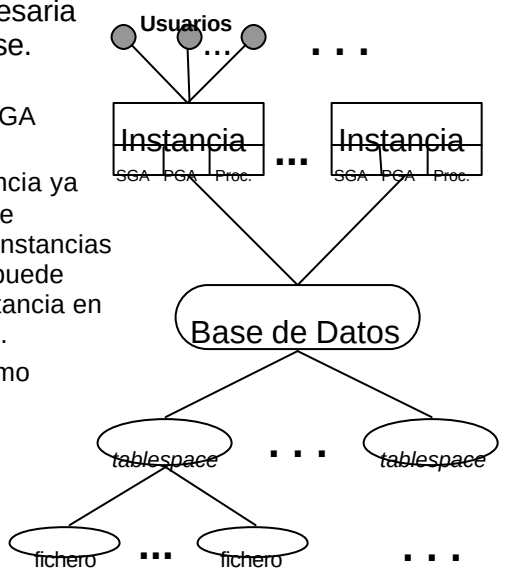
- **Estructuras básicas:**
  - **E. Física:** Almacenamiento de datos.
  - **E. Lógica:** Rep. de los datos y sus relaciones (esquema conceptual).
- **Estructura Lógica de una BD Oracle:**
  - **Objetos del esquema** (*schema objects*): Definición de tablas, vistas, índices, sinónimos, procedimientos almacenados...
  - **Espacios de Tablas** (*tablespaces*)**:** Es un área lógica de almacenamiento.
    - Informan cómo debe ser utilizado el espacio físico de la Bd.
    - Describen el almacenamiento físico, gestionando el espacio físico que usa la BD.
    - Cada BD tiene al menos un *tablespace*, aunque puede tener más para mejorar su gestión (uno para usuarios, aplicaciones, *rollback*...).
    - Cada *tablespace* pertenece sólo a una BD y se divide en 1 ó más ficheros de datos.
- **Estructura Física de una BD Oracle: Tipos de Ficheros (*datafiles*):**
  - **Datos:** Existen uno o más ficheros que contienen los datos actuales.
  - **Ficheros del Registro de Rehacer** (*redo log*)**:** Registran los cambios efectuados, para poder efectuar operaciones de recuperación (*recovery*).
  - **Ficheros de Control:** Información general, como nombre de la BD, nombres de sus ficheros, sus localizaciones, fecha de creación, histórico de *backups*...
  - **Ficheros para Rastrear** (*trace files*) y para **Registrar Alarmas** (*alert log*)**:** Se registran las operaciones por las que han pasado determinados procesos y los eventos importantes acaecidos a la BD.

Instancias de ORACLE (Oracle Instance)

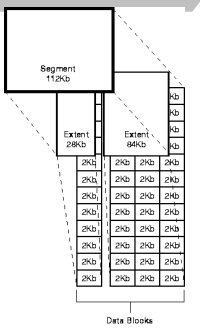
- **Instancia o Servidor de BD:** Conjunto de estructuras de memoria y procesos que acceden a los archivos de una BD. Distintas instancias pueden acceder a la misma BD.
  - ***System Global Area* (SGA):** Es un área de memoria con la información de la BD que pueden compartir los usuarios. Se crea cuando se empieza a usar una BD concreta. Puede verse alguna información desde SQL*PLUS, usando el comando **`SHOW SGA`**. Esta compuesta por:
    - **Caché de BD:** Con los bloques de BD más recientemente accedidos, para reducir los accesos a disco.
    - **Buffer del Registro de Rehacer,** para el fichero de *redo log*.
    - **Memoria compartida:** Para consultas SQL y otros procesos.
  - ***Program Global Area* (PGA):** *Buffer* de memoria con información sobre los procesos.
  - **Procesos de Usuario:** Aplicaciones que ejecuta el usuario.
  - **Procesos de Oracle:** Procesos del servidor (para atender a los usuarios...) y procesos de segundo plano (*background*), para tareas de registro, monitorización...

Iniciar/Finalizar ORACLE

- **Inicialización (*Startup*):** Es necesaria para que el SGBD pueda utilizarse. Pasos:
  - **Crear una Instancia:** Crear el SGA y los procesos de *background*.
  - **Montar una BD:** Asocia la instancia ya creada a una BD concreta. Puede asociarse la misma BD a varias instancias (modo compartido o paralelo) o puede exigirse que sea sólo en una instancia en cada momento (modo exclusivo).
  - **Abrir la BD:** Establece la BD como disponible para sus operaciones.
- **Finalización (*Shutdown*):** Es el proceso inverso:
  - **Cerrar la BD.**
  - **Desmontar la BD.**
  - **Borrar la Instancia Oracle.**

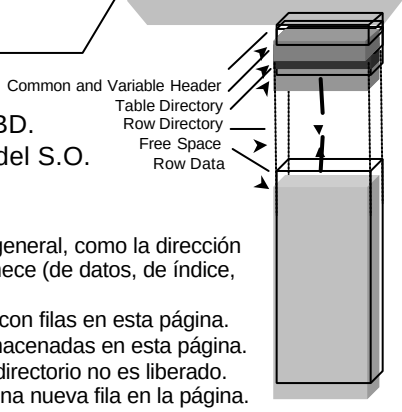


Páginas, Extensiones y Segmentos



- **Página o Bloque de Datos (*data blocks*):** Unidad mínima de asignación de espacio en la Base de Datos.
  - Es la menor unidad de E/S que puede utilizar la BD (independientemente de que el tamaño de bloque del S.O. sea menor).
- **Extensión (*extent*):** Conjunto de **páginas** *contiguas*, con un tipo de información específico.
- **Segmentos (*segment*):** Conjunto de **extensiones** que almacenan un determinado tipo de datos.
  - Al crear una estructura de datos, Oracle le asigna un **segmento** con una única **extensión**.
  - Cuanco se llena esa extensión se le asignan otras **extensiones** a ese **segmento**. Por eso, las extensiones no suelen ocupar espacios consecutivos, como sería deseable.
  - Un **segmento** completo se almacena en un *tablespace*, que puede distribuir sus **extensiones** en distintos ficheros.
    - Cada **extensión** se almacenará siempre en un único fichero.

Páginas o Bloques de Datos



- **Tamaño de Página:**
  - Especificado por el DBA cuando crea la BD.
  - Debe ser múltiplo del tamaño de bloque del S.O.
- **Formato de una Página:**
  - ***Overhead*:** Formado por 3 zonas:
    - ***Common and Variable Header*:** Información general, como la dirección de la página, el tipo de segmento al que pertenece (de datos, de índice, de rollback...).
    - ***Table Directory*:** Información sobre las tablas con filas en esta página.
    - ***Row Directory*:** Información sobre las filas almacenadas en esta página.
      - Al borrar una fila, su información en este directorio no es liberado. Ese espacio se vuelve a usar al insertar una nueva fila en la página.
  - ***Row Data*:** Zona con datos de tablas o índices. Una fila puede estar en varias páginas.
  - ***Free Space*:** Espacio libre para insertar nuevas filas (**`INSERT`**) o nuevos valores en las filas ya existentes (**`UPDATE`**), si requieren más espacio.
    - ***Transaction entries*:** También se guarda aquí información sobre las **transacciones** (**`INSERT`**, **`UPDATE`** y **`DELETE`**) sobre las filas de esta página. Se controla con dos valores que no es recomendable modificar:
      - **`INITRANS`:** Número inicial de entradas para las transacciones de esta página.
      - **`MAXTRANS`:** Máximo número de transacciones concurrentes para esta página.
    - Si el número de transacciones concurrentes supera **`INITRANS`**, Oracle guarda la información dinámicamente hasta que se excede **`MAXTRANS`** o hasta que no haya espacio libre en la página.

Páginas o Bloques de Datos (data blocks)

- **Control del Espacio Libre para Inserción y Actualización de Filas:** Existen dos parámetros que se especifican cuando se crea o altera una tabla o un índice:
  - **PCTFREE:** Mínimo porcentaje de página que se reserva como espacio libre para futuras actualizaciones de filas que ya existen en la página.
    - Por defecto es el 10%.
    - En una tabla relativamente estática un buen valor puede ser el 5%.
  - **PCTUSED:** Después de que una página sea considerada llena en función del límite especificado en el **PCTFREE**, Oracle no vuelve a introducir ninguna nueva fila en la misma hasta que el porcentaje de página ocupada sea menor que **PCTUSED**. Hasta entonces, el espacio libre sólo será dedicado a la actualización de las filas ya existentes en la página.
    - Por defecto deja el 40%, que indica que cuando esa página se llena no volverá a estar libre para inserciones hasta que tenga menos del 40% de ocupación (máximo 39%).
    - En una tabla relativamente estática un buen valor puede ser el 75%.
  - Por supuesto, **PCTFREE + PCTUSED <= 100**
  - Espacio libre para inserciones**:** **Tama_Página - Overhead - PCTFREE**
    - Cuando se produce un **INSERT**, Oracle mira la lista de páginas que están disponibles (*free list*) y selecciona la primera que encuentra.
  - Para actualizaciones (**UPDATE**) cualquier espacio libre puede ser utilizado.

Páginas o Bloques de Datos (data blocks)

- **Reagrupación del Espacio Libre en una Página:**
  - El **espacio libre** aumenta por las instrucciones **DELETE** o **UPDATE** (si la actualización establece valores que ocupan menos espacio).
  - Todo ese **espacio libre** podrá usarlo una instrucción **INSERT:**
    - Si la instrucción **INSERT** está en la misma transacción que la instrucción que ha generado el espacio libre (y situado después, naturalmente).
    - Si la instrucción **INSERT** está en otra transacción y la transacción que deja el espacio libre ya efectuó su **COMMIT**. Posiblemente ambas transacciones sean de distintos usuarios.
- **Encadenamiento y Migración de Filas:**
  - Los datos de una fila pueden ser demasiado grandes para caber en una página**:**
    - **La fila es muy grande:** Oracle almacena la fila en una cadena de páginas del mismo segmento.
      - Con datos grandes (como el tipo **LONG**) esta fragmentación es inevitable.
    - **Una fila es actualizada y sus nuevos valores no caben en su página actual:** Oracle traslada toda la fila a una nueva página, suponiendo que caben en una nueva página.
      - Oracle conserva la cabecera de la fila en su página inicial, apuntando a la nueva dirección en la nueva página. Así, el identificador de fila (**ROWID**) no cambia.
  - Esto hace que la eficiencia al tratar esta fila sea menor, ya que Oracle debe leer más de una página para recuperar la información de esa fila.

Extensiones (extents)

- **Un Segmento es un Conjunto de Extensiones.**
  - Si un segmento se llena, Oracle crea una nueva extensión para ese segmento (*incremental extent*) del mismo tamaño o superior.
- **Hay Dos Formas de Gestionar las Extensiones:**
  - **Extensiones Gestionadas Localmente (`LOCAL`):** Al crear un *tablespace* se pueden especificar las siguientes opciones:
    - **`AUTOALLOCATE`:** Son gestionadas por el sistema. Se especifica el tamaño de la extensión inicial y el tamaño del resto es calculado por Oracle, con un mínimo de 64KB.
    - **`UNIFORM`:** El tamaño especificado es para todas las extensiones (1MB por defecto).
  - **Extensiones Gestionadas por el Diccionario de Datos (`DICTIONARY`):** Utilizan como valores por defecto los valores almacenados en el Diccionario de Datos de la Base de Datos. Esos valores por defecto pueden modificarse en cualquier momento.
    - Estos valores son **`INITIAL`** (tamaño del primero), **`NEXT`** (tamaño del segundo) y **`PCTINCREASE`** (porcentaje de incremento en el tamaño del siguiente respecto al anterior).

Extensiones (extents)

- **Eliminar Extensiones:** En general, las extensiones de un segmento **no** son liberadas (*deallocated*) a no ser que borre el objeto almacenado en el segmento (mediante una instrucción **`DROP TABLE`** o **`DROP CLUSTER`**).
  - No obstante se producen algunas excepciones. Por ejemplo, el Administrador puede desasignar extensiones no utilizadas mediante la instrucción:

```
ALTER TABLE nombre_de_tabla DEALLOCATE UNUSED;
```

- **Tipos de Extensiones:** Según la información que almacenen.
  - **Tablas "*Nonclustered*" o No Agrupadas:** Son aquellas cuyas páginas asignadas solamente almacenan filas de esa tabla.
    - Incluso, si se borran todas las filas de una tabla, Oracle no reclama las páginas de datos para uso de otros objetos de la base de datos.
    - Tan sólo en el caso de que borre la tabla, este espacio podrá ser reclamado por otras extensiones que requieran de espacios libres.
  - **Tablas "*Clustered*" o Agrupadas:** Son aquellas tablas que almacenen su información en segmentos creados para un *cluster* y que, por tanto, pueden contener filas almacenadas en páginas junto a filas de otras tablas.
    - Si borramos la tabla con la instrucción **`DROP`**, el segmento y extensiones que ocupaba no se liberan ya que puede ser utilizado por las otras tablas del *cluster*.
  - **Índices:** Sus extensiones asignadas se liberan cuando se borra el índice o su tabla asociada.
  - **Otros:** *Snapshots* y sus *logs*, y de segmentos de Rollback o Temporales.

Segmentos (segments)

- **Un Segmento es un Conjunto de Extensiones** que contienen todos los datos de una estructura lógica específica (una tabla, un índice...) en un *tablespace***.**
- **Cuatro Tipos de Segmentos:**
  - **S. de Datos (*data segments*):** Se crea con la sentencia **`CREATE`** (para tablas "*nonclustered*", *snapshots*, *clusters*...). Los parámetros de almacenamiento (*storage parameters*) de páginas y extensiones se asignan con **`CREATE`** o **`ALTER`**, y afectan a la eficiencia en el almacenamiento y en la recuperación de datos.
  - **S. de Índices (*index segments*):** La sentencia **`CREATE INDEX`** crea un segmento también y pueden fijarse los parámetros de almacenamiento.
    - Una tabla y sus índices pueden tener segmentos en distinto *tablespace*.
  - **S. Temporales (*temporary segments*).**
    - Se explican a continuación.
  - **S. de *Rollback* (*rollback segments*).**
    - Se explican a continuación.

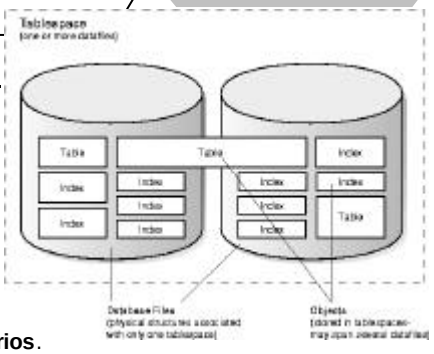
Segmentos Temporales

- **S. Temporales (*temporary segments*):** Cuando se procesa una consulta, Oracle requiere espacio temporal para realizar las operaciones intermedias de la instrucción SQL.
  - Para ello, automáticamente crea un espacio en disco: **Segmento Temporal.**
  - Operaciones que usan segmentos temporales: **`CREATE INDEX`**, **`SELECT...ORDER BY`**, **`SELECT DISTINCT`**, **`SELECT...GROUP BY`**, **`SELECT...UNION`**, **`SELECT...INTERSECT`** y **`SELECT...MINUS`**.
    - Normalmente solo se requiere este espacio cuando se necesita ordenar un resultado que, además, no cabe en memoria.
  - Como máximo se usarán **dos** segmentos temporales.
  - **Oracle** crea los segmentos temporales que necesita durante una sesión de usuario en el ***tablespace* temporal del usuario** que realiza la instrucción.
    - Este *tablespace* es especificado en la instrucción **`CREATE USER`** o, posteriormente, con **`ALTER USER`** y usando la opción **`TEMPORARY *TABLESPACE*`**.
    - Si el Administrador no ha definido estos parámetros, por defecto el *tablespace* temporal será el ***tablespace* SYSTEM**.
  - Una vez la instrucción se completa, **Oracle borra el segmento temporal** asignado.
  - **Es razonable Crear un *Tablespace* especial para contener los segmentos temporales** debido a que la creación y borrado de segmentos temporales ocurre frecuentemente. De esta forma puede distribuir las entradas/salidas por distintos discos mejorando los tiempos de respuesta.

Segmentos de Rollback

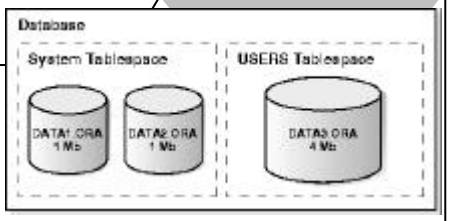
– **S. de *Rollback* (*rollback segments*):** En estos segmentos se almacenan los viejos valores de los datos que han sido modificados por las transacciones. Su objetivo es mantener la consistencia, realizar *rollback*s y utilizarse en operaciones de recuperación (*recovery*) de la BD.
  - Cada BD contiene **uno o más segmentos de *rollback*.**
  - La **información** que contiene un segmento de *rollback* consiste en numerosos registros o *rollbacks entries*.
    - **Ejemplo:** Incluye información sobre el dato modificado (fichero *filenumber* y página *block ID*), así como el contenido de éste (si existía antes de realizar la operación).
  - **Oracle** enlaza cada entrada de una **transacción** de manera que todas ellas son fácilmente localizables en caso de tener que deshacerla.
  - **El propietario de estos segmentos es el usuario SYS**, por lo que nadie, incluido el Administrador, puede acceder a ellos.
  - Al grabar las **entradas de *rollback***, cambian las páginas del segmento de *rollback* y Oracle guarda todos los cambios en las páginas, incluyendo estas entradas de *rollback*, en el ***Redo Log*** (registro de rehacer).
    - Este segundo almacenamiento de la información de *rollback* es muy importante para las transacciones activas (sin terminar con **`COMMIT`** o **`ROLLBACK`**).
    - **Si el sistema cae**, Oracle automáticamente restaura la información de los segmentos de *rollback*, incluyendo las entradas para las transacciones activas. Tras eso, Oracle realiza *rollback*s de las transacciones que estaban activas en el momento de la caída del sistema.
    - Cuando se realiza un **`COMMIT`**, Oracle libera la información de *rollback*.

Tablespaces y Datafiles



- **Oracle almacena los datos lógicamente en *tablespaces* (espacios de tablas) y físicamente en *datafiles* (ficheros).**
- ***Tablespaces*:**
  – Una **BD** está dividida en **uno o más *tablespaces*** (áreas lógicas de almac.).
  – El **Administrador** los usa para:
    - Controlar la **creación de espacios en disco** para los datos de la BD.
    - Asignar **cuotas específicas para los usuarios**.
    - Controlar la **accesibilidad** de los datos (poniendo un *tablespace* en modo *online*/*offline* o *read-only*/*read-write*).
    - Realizar operaciones parciales de ***backups*/*restore***.
    - Repartir los datos en **varios discos** para mejorar el **rendimiento**.
  – El **Administrador** puede: crear y borrar *tablespaces*, añadir ficheros a los *tablespaces*, añadir o alterar segmentos del *tablespace*, hacer que un *tablespace* sea temporal o permanente...
  – Podemos **Crear *Tablespaces*** temporales o no con la instrucción SQL:
    `CREATE *TABLESPACE tablespace* TEMPORARY / PERMANENT;`
  – También podemos **Cambiar** el estado de un ***Tablespace*** con:
    `ALTER TABLESPACE *tablespace* TEMPORARY;`

Tablespaces y Datafiles



- **El *Tablespace* SYSTEM:** Cada BD Oracle contiene un *tablespace* llamado **SYSTEM**, que ORACLE crea automáticamente al crear la BD.
  - Contiene las **tablas del Diccionario de Datos** para toda la BD.
  - Una BD pequeña puede necesitar sólo el *tablespace* **SYSTEM**.
    - Sin embargo, es recomendable crear, al menos, un *tablespace* adicional a fin de separar la información del diccionario de los datos.
      - Esto da más flexibilidad en las tareas de administración y reduce los problemas del acceso concurrente al mismo *tablespace* (contención).
  - Evidentemente, el *tablespace* **SYSTEM** está siempre *online* mientras la base de datos esté abierta.
- ***Tablespaces Read-Only*:** La principal ventaja de los *tablespaces read-only* consiste en eliminar la necesidad de realizar *backups* y recuperaciones de porciones de la base de datos.
  - **Oracle** nunca actualiza los ficheros de un *tablespace read-only* y, por tanto, estos *tablespaces* pueden residir en **dispositivos de sólo lectura**, como CD-ROM.
  - Se realiza mediante la instrucción **ALTER TABLESPACE**.

Tablespaces y Datafiles

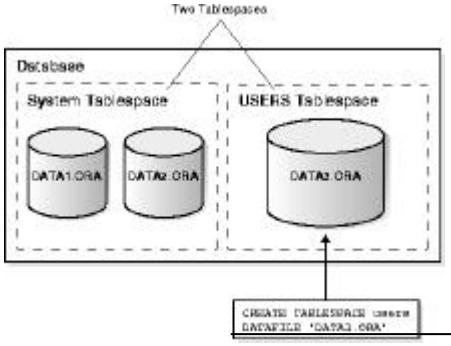
- ***Tablespaces Online* y *Offline*:** Un Administrador de la BD puede poner a los *tablespaces* accesibles o no, mientras la BD está abierta.
  - Normalmente, un *tablespace* está ***online*** para que los usuarios tengan accesibles los datos. Los motivos por los que un Administrador puede poner ***offline*** a un *tablespace* pueden ser:
    - Para dejar inaccesible una porción de la BD mientras el resto sigue accesible.
    - Para realizar un ***backup offline*** del *tablespace.*
    - Para hacer que algunas aplicaciones y algunas tablas queden temporalmente inaccesibles al objeto de realizar operaciones de mantenimiento o modificación.
  - **No** se pueden poner ***offline*** *tablespaces* que contengan segmentos de *rollback* que estén en uso, ni el *tablespace* **SYSTEM**.
  - Oracle puede poner un *tablespace* ***offline*** si se ha producido algún **error** importante (como un error de disco...).
- ***Tablespaces* Temporales:**
  - Se puede utilizar espacio para la realización de operaciones de ordenación (*sort*) de forma más eficiente creando *tablespaces* temporales para este exclusivo uso.
    - Un *tablespace* temporal puede ser utilizado únicamente para contener segmentos *sort*.
    - Objetos permanentes no pueden residir en un *tablespace* temporal.

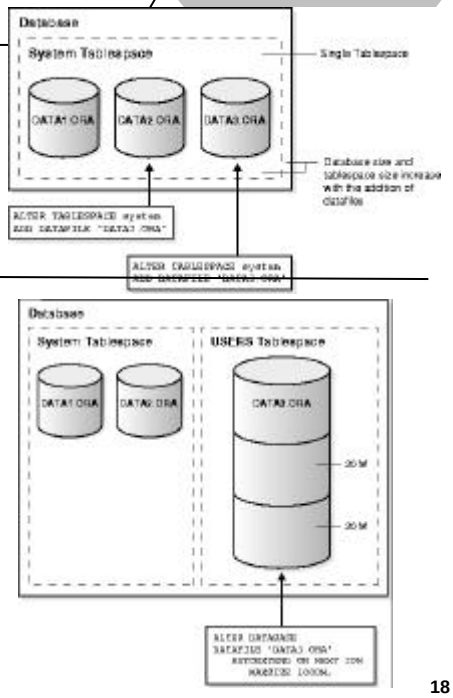
Tablespaces y Datafiles

- Un ***tablespace*** de una BD consiste en uno o más ***datafiles***:
  - Un *datafile* puede estar asociado con un único *tablespace*.
  - El **primer *tablespace*** en cualquier BD Oracle siempre es el *tablespace* **SYSTEM** y para éste se construye el primer *datafile* cuando creamos la base de datos.
- **Contenido del *Datafile*:**
  - Cuando un *datafile* se crea, su espacio es formateado para que pueda contener los datos de usuario.
  - Los datos asociados con los objetos del esquema serán almacenados físicamente en *datafiles*, pero debe tenerse en cuenta que no existe una correspondencia directa entre estos objetos y los *datafiles*.
    - Un **objeto** no se corresponde con un ***datafile***, sino con un ***tablespace***.
- **Tamaño de los *datafiles*:**
  - Podemos alterar el tamaño de un *datafile* después de haber sido creado.
  - Esto permite facilitar las tareas de administración de la base de datos.
  - La instrucción correspondiente está dentro de **ALTER DATABASE**.

Tablespaces y Datafiles

- **Podemos hacer Crecer la BD de tres maneras:**
  1. Añadiendo un *datafile* al *tablespace:*
  2. Añadiendo un nuevo *tablespace:*





  3. Incrementando el tamaño de un *datafile:*

Algunas Estructuras Internas de las BD

- **Tablas, Columnas** y **Restricciones** (**`TABLE`**, **`COLUMN`**, **`CONSTRAINT`**)**:** Mecanismo de almacenamiento en BD relacionales. Las columnas tienen su tipo de dato asociado.
- **Vistas** (**`VIEW`**)**:** No se almacenan sus datos, sino su definición.
- **Usuarios** (**`USER`**)**:** Que son los propietarios de los distintos objetos.
- **Índices** (**`INDEX`**)**:** Estructuras para que permite localizar rápidamente una fila particular de una tabla. Utiliza el **`ROWID`**, un identificador que Oracle asigna a cada fila y que indica el lugar exacto de dicha fila.
- **Grupos** (**`CLUSTER`**)**:** Son grupos de tablas que se almacenan juntas porque se suelen acceder conjuntamente y, así, se consigue mejor eficiencia.
- **Secuencias** (**`SEQUENCE`**)**:** Es un mecanismo para obtener una lista consecutiva de números exclusivos, para simplificar las tareas de programación. Pueden ser cíclicas (o no) y tener un incremento cualquiera (1 es lo normal).
- **Procedimientos y Funciones** (**`PROCEDURE`**, **`FUNCTION`**)**:** Sentencias escritas en un lenguaje procedural (como PL/SQL, java...) y que pueden utilizar las aplicaciones. Proporcionan mecanismos de seguridad, ya que puede evitarse que los usuarios accedan directamente a ciertas tablas.
- **Paquetes** (**`PACKAGE`**)**:** Grupos de procedimientos, funciones y/o variables.
- **Disparadores** (**`TRIGGER`**)**:** Procedimientos que se ejecutan cuando ocurre algún evento (**`INSERT`**, **`UPDATE`** o **`DELETE`**).
- **Sinónimos** (**`SYNONYM`**)**:** Simplifican el uso de nombres largos (propietario.tabla).
- **Roles** (**`ROLE`**)**:** Los roles son conjuntos de privilegios que pueden concederse 'de golpe' a un usuario. Los **Privilegios** pueden ser de **Objetos** (para **`INSERT`**, **`SELECT`**, **`UPDATE`**, **`DELETE`**, **`EXECUTE`**...) o del **Sistema** (para crear tablas, vistas...).
- **Perfiles** (**`PROFILE`**)**:** Características genéricas para cierto grupo de usuarios (número máximo de sesiones, si caduca la clave, días que faltan para caducar su cuenta...).

Diccionario de Datos de ORACLE

- **Diccionario de Datos** *(data dictionary)**:*** Es un conjunto de tablas de sólo lectura con los metadatos (descripción del esquema).
- **Información que contiene el Diccionario:**
  - La definición de todos los objetos o estructuras internas del esquema (tablas, vistas, índices, clusters, sinónimos, secuencias, procedimientos, paquetes, funciones, triggers...).
  - Cuánto espacio está reservado o tienen ocupado los objetos del esquema.
  - Los valores por defecto de las columnas.
  - Información sobre las restricciones de integridad.
  - Nombres de los usuarios de Oracle.
  - Privilegios y Roles que los usuarios tienen concedidos.
  - Información sobre auditoría de la BD.
  - Otras informaciones de carácter general (estadísticas...).
- **Sólo ORACLE debe Escribir y Leer en las Tablas del Diccionario:**
  - Oracle accede cada vez que se ejecuta una **sentencia DDL**.
  - Los usuarios podrán acceder a algunas **vistas**: En general, no se deben modificar estas vistas.
  - Oracle también accede para hacer comprobaciones cuando lo necesita.

Diccionario de Datos de ORACLE

- **Estructura del Diccionario:**
  - **Tablas del Diccionario:** Son propiedad del usuario **SYS**.
    - Esas tablas no son muy útiles directamente porque están en un formato muy críptico.
    - La Tabla **DUAL:** Es una pequeña tabla del Diccionario de Datos que utiliza Oracle y los usuarios para realizar algunas operaciones. Contiene tan sólo una columna llamada **DUMMY** y una fila conteniendo el valor '**X**'.
  - **Vistas del Diccionario** (sobre esas tablas)**:** Pertenecen a **SYSTEM**.
    - Pueden ser utilizadas por el resto de los usuarios de la BD.
    - Existen sinónimos públicos para acceder fácilmente a ellas.
- **Prefijos en las Vistas del Diccionario de Datos:**

Cada vista obtiene un subconjunto de datos de la siguiente

  - **USER_** Objetos que pertenecen al propio usuario.
    - No tienen la columna **OWNER**, ya que el propietario es el usuario actual.
  - **ALL_** Todos los objetos accesibles por el usuario.
    - Accesibles por el usuario por concesión explícita o pública de privilegios o roles.
  - **DBA_** Todos los objetos existentes (sólo para los DBA).
    - Accesibles para los usuarios con el privilegio **SELECT ANY TABLE**.

Vistas del Diccionario de Datos

Probar también cambiando el prefijo.

- **Algunas vistas** con el prefijo **USER_** (pueden verse en **ALL_VIEWS**)**:**
  - **USER_OBJECTS:** Lista de todos los objetos pertenecientes al usuario (tablas, vistas, paquetes, índices, *triggers*, sinónimos...).
  - **USER_TABLES:** Lista de todas las tablas del usuario.
  - **USER_VIEWS:** Vistas del usuario.
  - **USER_USERS:** Diversos datos sobre el usuario.
  - **USER_UPDATABLE_COLUMNS:** Columnas que pueden ser modificadas.
  - **USER_JOBS:** Tareas pertenecientes al usuario.
  - **USER_TRIGGERS:** Disparadores (*triggers*) del usuario.
  - **USER_SYNONYMS:** Sinónimos pertenecientes al usuario.
  - **USER_INDEXES:** Índices pertenecientes al usuario.
  - **USER_CONSTRAINTS:** Restricciones pertenecientes al usuario.
    - **USER_CONS_COLUMNS:** Columnas involucradas en cada restricción.
  - **USER_TAB_PRIVS:** Permisos sobre objetos con el usuario involucrado. Si pone **_COL_** en vez de **_TAB_** se refiere a las columnas. Se puede distinguir entre:
    - **USER_TAB_PRIVS_MADE:** Permisos sobre los objetos del usuario.
    - **USER_TAB_PRIVS_RECD:** Permisos recibidos por el usuario.
  - **USER_TAB_COLUMNS:** Descripciones de las columnas del usuario.
  - **USER_TAB_COMMENTS y USER_COL_COMMENTS :** Comentarios sobre las tablas y columnas del usuario,si se han insertado con el comando **COMMENT:**

```
COMMENT ON [TABLE|COLUMN] <Tabla>[.<Columna>] IS '<Texto>';
```

Operaciones sobre el Dicc. de Datos

Buscar y Visualizar la Llave Primaria:

Procedimiento **PL/SQL** que efectúa esa operación sobre una tabla dada en su único argumento.

Ejemplo de utilización: `EXEC PRIMARIA ('PRODUCTOS');`

```
/*****************************************************************/
/* PROCEDIMIENTO PARA MIRAR CLAVE PRIMARIA DE UNA TABLA DADA  */
CREATE OR REPLACE PROCEDURE PRIMARIA
  (Var_nombre_tabla IN user_constraints.table_name%type) IS

var_primaria user_constraints.constraint_name%type;
var_campo    user_cons_columns.column_name%type;

CURSOR busca_campos IS
   SELECT column_name FROM user_cons_columns
   WHERE  constraint_name=var_primaria;
BEGIN
   dbms_output.put_line('***** ***** Clave Primaria ***** *****');

   /* Calcular Nombre de la restricción PK */
   SELECT constraint_name INTO var_primaria
   FROM   user_constraints
   WHERE  table_name=Var_nombre_tabla
     AND  constraint_type='P';

   dbms_output.put('* '||var_primaria||': ');

   OPEN busca_campos;
   LOOP
     FETCH busca_campos INTO var_campo;
     EXIT WHEN busca_campos%notfound;
     dbms_output.put(var_campo||',');
   END LOOP;
   CLOSE busca_campos;
   dbms_output.new_line;
EXCEPTION
  WHEN no_data_found THEN dbms_output.put_line('* NO TIENE PK *');
END PRIMARIA;
/
```


Operaciones sobre el Dicc. de Datos

Buscar y Visualizar las Llaves Externas o Foráneas: Procedimiento **PL/SQL** que efectúa esa operación sobre una tabla dada en sus argumentos.

Ejemplo de utilización: `EXEC FORANEAS ('PRODUCTOS');`

```
set serveroutput ON size 20000;

/*************************************************/
/* MIRAR CLAVES FORÁNEAS DE UNA TABLA DADA en ARGS. */
CREATE OR REPLACE PROCEDURE FORANEAS
  (Var_nombre_tabla IN
                 user_constraints.table_name%type) IS

var_restriccion user_constraints.constraint_name%type;
var_references
             user_constraints.r_constraint_name%type;
var_tabla_ref user_constraints.constraint_name%type;
var_column1   user_cons_columns.column_name%type;
var_column2   user_cons_columns.column_name%type;

CURSOR busca_foranea IS
   SELECT constraint_name,r_constraint_name
   FROM   user_constraints
   WHERE  table_name=Var_nombre_tabla
     AND  constraint_type='R';

CURSOR busca_campos IS
   SELECT u1.column_name, u2.column_name
   FROM   user_cons_columns u1, user_cons_columns u2
   WHERE  u1.position = u2.position
     AND  u1.constraint_name=var_restriccion
     AND  u2.constraint_name=var_references;
BEGIN
```

```
  dbms_output.put_line
         ('***** Llaves Externas o Foráneas *****');

  OPEN busca_foranea;
  LOOP /* RECORRE LAS FORÁNEAS DE LA TABLA DADA*/
    FETH busca_foranea INTO var_restriccion,
                            var_references;
    EXIT WHEN busca_foranea%notfound;
    dbms_output.put('* '||var_restriccion||' ---- FK hacia ');

    /* BUSCA TABLA BASE DE LA FORANEA*/
    SELECT table_name INTO var_tabla_ref
    FROM user_constraints
    WHERE constraint_name = var_references;
    dbms_output.put_line(var_tabla_ref||' (R_CONSTRAINT_NAME '
                         ||var_references||')');

    OPEN busca_campos;
    LOOP /* Busca los campos que corresponden a cada uno */
      FETCH busca_campos INTO var_column1, var_column2;
      EXIT WHEN busca_campos%notfound;
      dbms_output.put_line('------------> '
        ||Var_nombre_tabla||'.'||var_column1||' --> '
        ||var_tabla_ref   ||'.'||var_column2);
    END LOOP;
    CLOSE busca_campos;
  END LOOP;
 CLOSE busca_foranea;
END FORANEAS;
/
```

Diccionario de Datos de ORACLE

- **Tablas de Ejecución Dinámica:**
  - **Tablas virtuales** que almacenan información sobre la actividad de la BD durante su funcionamiento.
  - **No son verdaderas tablas** y no son accesibles para la mayoría de los usuarios. Sin embargo, los Administradores de la BD pueden consultar y crear **vistas** sobre esas tablas (llamadas *fixed views*) y autorizar el acceso a esas vistas por parte de otros usuarios.
  - El propietario de estas tablas es **SYS** y los nombres de las mismas comienzan por **V_$**. Para estas tablas se crean unas vistas y para las vistas se crean sinónimos que comienzan todos por **V$**.
  - **Ejemplos:**
    - **V$DATABASE** contiene información sobre la BD (nombre...).
    - **V$DATAFILE** contiene información sobre los *datafiles*.
    - **V$FIXED_TABLE** contiene información sobre todas las tablas de ejecución dinámica y vistas de la BD.
    - **V$SESSION** contiene información sobre las sesiones actuales (usuario, comando actual, programa y terminal de acceso...).
    - **V$PROCESS** contiene información sobre los procesos activos.
    - **V$CONTROLFILE** contiene una lista con los ficheros de control.

Estructura de la Memoria en ORACLE

- **Oracle Utiliza la Memoria para Almacenar:**
  - Código del programa que se ejecuta.
  - Información sobre las sesiones conectadas.
  - Información necesaria sobre las ejecuciones de los programas (estados de las consultas...).
  - Información compartida entre procesos (sobre los bloqueos...).
  - Caché de datos.
- **Estructuras Básicas de Memoria** de una **instancia** de Oracle:
  - **Área Global del Sistema (SGA, System Global Area).**
  - **Áreas Globales de Programas (PGA, Program Global Areas).**
  - **Área de Ordenaciones (Sort Areas).**
  - **Memoria Virtual.**
  - **Áreas de Código de *Software* (SCA, Software Code Areas).**

Estructura de la Memoria: SGA

- **Área Global del Sistema (SGA, System Global Area).**
  - Un **SGA** es un grupo de estructuras de memoria compartida que contienen **datos** e **información de control de una Instancia** de una BD.
  - Si a una **Instancia están conectados múltiples usuarios** concurrentes, los datos de éstos se **comparten** en el **SGA**.
  - El **SGA** se **crea** cuando se **arranca una Instancia** de una BD y se **destruye** cuando se **cierra la Instancia**.
  - El **SGA es de lectura y escritura**, y contiene las siguientes estructuras de datos:
    - **Caché de los *Buffers*** de la BD (*Database Buffer Cache*).
    - ***Buffer* del Dietario o del Registro de Rehacer** (*Redo Log Buffer*).
    - **El "Pool" Compartido** (*Shared Pool*).
      - **Caché de Biblioteca.**
      - **Caché del Diccionario de Datos.**
      - **Estructuras de Control.**
    - **Varias informaciones diversas.**
  - **SGA fijo** (*fixed* SGA)**:** Parte del **SGA** con información general sobre el estado de la BD y la Instancia, a la que los procesos de "*background*" necesitan acceder
    - Ningún dato de usuario se almacena aquí.

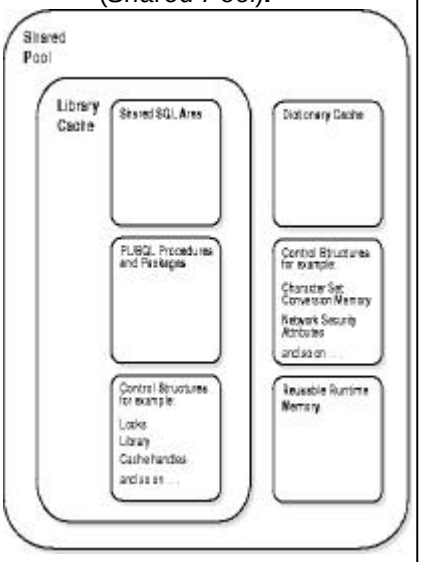
Estructura de la Memoria: SGA

- **Caché de los *Buffers*** de la BD (*Database Buffer Cache*).
  - Es la parte del **SGA** que contiene copias de las **páginas leídas** de los *datafiles*. Todos los procesos de usuarios que están conectados a la Instancia comparten el acceso a esta caché.
  - Los ***buffers* en la caché** están organizados en dos listas:
    - **Lista "en espera" (*write list*):** Contiene *buffers* en espera (*dirty buffers*) en el sentido de que contienen datos que han sido modificados pero que aún no han sido escritos a disco.
    - **Lista LRU (menos recientemente usada, Least Recently Used):** La lista LRU contiene *buffers* libres, *buffers* que están siendo accedidos actualmente (*pinned buffers*) y, por último, los *buffers* "en espera" que aún no han sido movidos a la lista "en espera".
  - La primera vez que un proceso requiere un dato, **se busca en la caché:**
    - **Si se encuentra** el dato en uno de estos *buffers*, se lee directamente de la memoria (*cache hit*), acelerando el proceso de lectura.
    - **Si no se encuentra** (*cache miss*), entonces debe obtenerse una copia de la página en disco y pasarla a un buffer de la caché antes de proceder a leerlo.

Estructura de la Memoria: SGA

- ***Buffer* del Dietario o del Registro de Rehacer** (*Redo Log Buffer*)**:**
  - El buffer del "*Redo Log*" o Dietario, es un **buffer circular** en el **SGA** que contiene información sobre los cambios que se producen en la BD.
  - Las entradas (*redo entries*) contienen información necesaria para reconstruir o **rehacer los cambios** efectuados a la BD mediante las **`INSERT`**, **`UPDATE`**, **`DELETE`**, **`CREATE`**, **`ALTER`** o **`DROP`**.
  - Estas entradas son copiadas por el servidor de procesos de Oracle desde el espacio de memoria del usuario hasta estos ***buffers* del SGA**, donde los registros van escribiéndose de forma secuencial.
    - Ahí esperan a que el proceso de *background* **LGWR** (*Log Writer*) escriba el buffer al correspondiente fichero activo (*online*) de "*Redo Log*" en disco.

El "Pool" Compartido (*Shared Pool*)**:**



Estructura de la Memoria: SGA

- **El "Pool" Compartido** (*Shared Pool*): Su tamaño total está determinado por el parámetro de inicialización **`SHARED_POOL_SIZE`**. Contiene:
  - **Caché de Biblioteca** (*Library Cache*)**:** Incluye la siguiente información:
    - **Áreas compartidas y privadas de SQL.**
      - Oracle asigna a cada instrucción de SQL que ejecuta, un área de SQL compartida y un área de SQL privada.
      - Oracle reconoce cuando dos usuarios están ejecutando la misma declaración de SQL y reutiliza el área de SQL compartida para esos usuarios:
        - » Se ahorra tiempo y memoria usando el área de SQL compartida, sobre todo en sistemas con muchos usuarios ejecutando la misma aplicación.
      - Sin embargo, cada usuario debe tener una copia separada de la instrucción en el área de SQL privada.
    - **Procedimientos y paquetes PL/SQL.**
      - Evitan tener varias copias de cada programa PL/SQL, aunque una parte es privada para cada usuario.
      - Las sentencias SQL contenidas dentro de un programa utilizan también las áreas SQL compartidas y privadas.
    - **Estructuras de control** (caché de gestión de bloqueos, de librerías...).
  - **Caché del Diccionario** (*Dictionary Cache*)**:** La continua necesidad de acceso al diccionario hace que Oracle disponga de dos localizaciones en memoria compartidas por todos los procesos, para contener esos datos:
    - Un área es ésta, en la que los datos se almacenan como filas.
    - La otra es la anteriormente vista de la caché de biblioteca.
  - **Estructuras de Control** (*Control Structures*)**:** Juegos de caracteres, sistemas de conversión, atributos de seguridad...

Estructura de la Memoria: PGA

- **Áreas Globales de Programa (PGA, Program Global Areas).**
  - Un **PGA** es una región de memoria que contiene datos e información del control para un solo proceso (de servidor o de *background*).
    - Suele llamarse "*Process Global Area*".
  - Un **PGA** es un área de memoria no compartida en la que puede escribir un proceso.
  - Para cada proceso del servidor se asigna un **PGA**.
    - Ese **PGA** es exclusivo para ese proceso
    - Ese **PGA** se lee y se escribe por Oracle, que actúa en nombre de ese proceso.
  - Un **PGA**, pues, es asignado por Oracle cuando un usuario se conecta a una base de datos y crea una sesión.
  - Un **PGA** siempre contiene un espacio que contiene las variables de la sesión, y alguna otra información. Este espacio es denominado *stack space*.
  - El tamaño de un **PGA** depende del sistema operativo específico.

Estructura de la Memoria en ORACLE

- **Área de Ordenaciones (Sort Areas):**
  - Ordenar filas siempre requiere espacio en memoria. Las zonas de la memoria en las que Oracle ordena los datos son denominadas áreas de ordenación.
  - El tamaño por defecto, expresado en bytes, es específico de cada S.O.
  - Si los datos a ordenar no caben en el Área de Ordenaciones, los datos se dividen en trozos que sí quepan, se ordenan y se mezclan (*merge*).
- **Memoria Virtual:**
  - **Oracle** puede usar la **memoria virtual** que ofrezca el S.O., simulando memoria a base de utilizar un almacenamiento secundario (discos...).
- **Área de Código de *Software* (SCA):**
  - Son zonas de memoria destinadas a **almacenar el código de Oracle** en ejecución o que puede ejecutarse. Este código de Oracle se almacena en una zona distinta, y más protegida, que las zonas dedicadas a almacenar los códigos de programas de usuarios.
  - Son **áreas de sólo lectura** de tamaño estático que solamente cambian cuando el *software* se instala y su tamaño depende del S.O.
  - **Pueden ser compartidas o no compartidas:** Las primeras son más eficientes porque el mismo código puede ser usado por distintos usuarios.

Estructura de los PROCESOS

- **Todo usuario** que se conecta a Oracle debe **ejecutar dos módulos** de código para acceder a la instancia de la BD Oracle, que se ejecutan como procesos normales bajo el control del S.O.**:**
  - **Aplicación o Herramienta Oracle:** Se trata de una aplicación normal que se conecte a Oracle, o bien una herramienta de Oracle (como Oracle Enterprise Manager o SQL*Plus).
    - Estos programas permiten ejecutar sentencias SQL.
  - **Código del Servidor de Oracle:** Es una parte de Oracle que se ejecuta para el usuario y que interpreta y procesa las órdenes SQL.
- **Sistemas Oracle Multiproceso:** Distintos procesos ejecutan distintas partes de Oracle, además de los procesos de los usuarios.
  - La mayoría de los SGBD son **multiusuario**, ya que es una de sus principales ventajas.
  - Dividiendo el trabajo en **distintos procesos** se consigue un mejor rendimiento dando servicio a múltiples usuarios y aplicaciones.
- **Tipos de Procesos:**
  - Procesos de Usuario (*User Processes*).
  - Procesos de Oracle (Oracle Processes): **1.** P. de Servidor (*server*) / **2.** P. en 2º Plano, o de Fondo (*background*)

Estructura de los PROCESOS

- **Procesos de Usuario:** Cuando el usuario ejecuta una aplicación, o una herramienta de Oracle, se crea un proceso de usuario.
  - **Conexión:** Es una vía de comunicación entre un proceso de usuario y una Instancia. Establece el mecanismo de comunicación.
  - **Sesión:** Es una conexión específica de un usuario a una Instancia a través de un proceso de usuario.
    - Por ejemplo, un usuario establece una **conexión** usando SQL*Plus, después introduce su *username* y *password* y, posteriormente, inicia una **sesión** (a través de la ejecución de un proceso de usuario).
    - Un mismo usuario puede tener varias sesiones.
- **Procesos de Oracle:** Pueden ser de 2 tipos:
  - **1. Procesos de Servidor:** Se crean para cada una de las aplicaciones de usuario que ejecutan una o más de las siguientes acciones:
    - Instrucciones SQL a través de aplicaciones.
    - Lectura de páginas de los *datafiles* en disco a los *buffers* compartidos de la BD en el SGA, si los bloques no se encuentran ya en el SGA.
    - Devolución de los resultados de forma que la aplicación pueda procesar la información.

Estructura de los PROCESOS

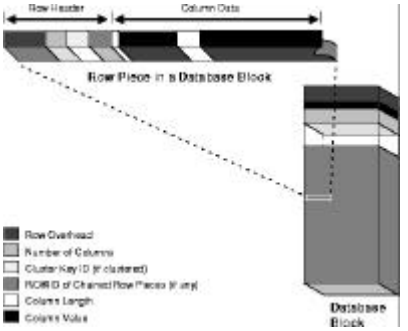
- **2. <u>Procesos de *Background*</u>:** Para maximizar el rendimiento y posibilitar el acceso simultáneo de múltiples usuarios, Oracle utiliza algunos procesos adicionales denominados de "*background*" (en 2º plano, o de fondo).
  - Algunos de estos procesos son creados automáticamente cuando se inicia una **Instancia**. No todos ellos están siempre presentes.
  - Estos **procesos** son los siguientes:
    - **Database Writer (DBW0 o DBWn):** Escribe el contenido de los *buffers* que han sido modificados a los *datafiles*.
    - **Log Writer (LGWR):** Es el responsable del funcionamiento de los *buffers* del *Redo Log* mediante la escritura de su contenido al fichero de *Redo Log*.
    - **Checkpoint (CKPT):** Cuando se realiza un *checkpoint*, Oracle actualiza las cabeceras de todos los *datafiles* que almacenan datos a causa de este *checkpoint*.
      - » Esta tarea incluye la escritura física de las páginas: Esto lo realiza el DBWn.
    - **System Monitor (SMON):** Este proceso tiene dos funciones:
      - » Maneja la recuperación de la BD a partir del fallo de una Instancia (esto es, cuando las estructuras de memoria y los procesos que componen la Instancia no pueden continuar por algún motivo).
      - » Chequea periódicamente los espacios de disco para determinar si une pequeños fragmentos de espacios libres.

Estructura de los PROCESOS

- **Process Monitor (PMON):** Actúa cuando falla un proceso de usuario, liberando los recursos que tuviera asignados (memoria...) y deshaciendo los cambios que hubiese realizado desde su último **`COMMIT`**.
- **Recoverer (RECO):** Es un proceso de *background* opcional para las BD distribuidas que gestiona las transacciones distribuidas.
- **Archiver (ARCH):** Es opcional. Copia los ficheros del *Redo Log* a un dispositivo predeterminado cuando éste está lleno. Este proceso solamente se presenta cuando el *Redo Log* se usa en modo **`ARCHIVELOG`** y el archivo automático está activado.
- **Lock (LCKn):** Opcional. Solamente para servidores paralelos. Gestiona los bloqueos entre distintas Instancias.
- **Job Queue (SNPn):** Opcional, para BD distribuidas.
- **Queue Monitor (QMNn):** Opcional. Monitoriza el orden de salida de los mensajes.
- **Dispatcher (Dnnn):** Opcional. Permite a los procesos de usuario compartir procesos del servidor, de manera que el servidor pueda soportar un mayor número de usuarios.
- **Shared Server (Snnn):** Cada uno de estos procesos sirve las peticiones de múltiples clientes en configuraciones como la anterior, compartiendo los procesos del servidor.

Objetos del Esquema: TABLAS

- **TABLAS (*tables*):** Son la unidad básica de almacenamiento de una BD Oracle.
  - Cuando se **crea una tabla**, Oracle automáticamente asigna un segmento de datos de un *tablespace* para contener los futuros datos de la tabla. Podemos controlar la generación de este espacio a partir de los parámetros correspondientes (**PCTFREE**, **PCTUSED**...).
  - **Oracle** almacena habitualmente cada **fila** de una tabla en una única página (***row piece***)**:**
    - **Una fila** se fragmenta en varias páginas (varios ***row pieces***) si por algún motivo la fila no puede ser almacenada completamente en la página: Se usan varias páginas, encadenándolas.
    - **Cada fila** de una tabla se almacena en dos partes: una que contiene información de cabecera (***row header***) y otra con el contenido de los datos (***column data***).



  - ***Row header:*** En una fila de un único bloque ocupa 3 bytes como mínimo.
  - ***Column data*:** Almacena la longitud de cada columna y sus datos.
    - El valor **NULL** se representa almacenando cero en la longitud y nada en los datos.
  - Borrar una columna es costoso, pero si se usa la opción **SET UNUSED** de **ALTER TABLE**, sólo marca la columna como borrada posponiendo el borrado.

Objetos del Esquema: Otras TABLAS

- **TABLAS Anidadas (*nested tables*):** Tablas con una columna de tipo tabla.
- **TABLAS Temporales (*temporary tables*):** Son tablas temporales que duran mientras dure la **transacción o sesión** actuales.
  - Sentencia de creación:

```
CREATE GLOBAL TEMPORARY TABLE <tabla>
  ON COMMIT [DELETE|PRESERVE] ROWS;
```

  - Debe especificarse si son datos específicos de la **transacción** o de la **sesión:**
    - **`ON COMMIT DELETE ROWS`:** Borra el contenido de la tabla al efectuar un **`COMMIT`**, es decir, los datos son específicos de la transacción.
    - **`ON COMMIT PRESERVE ROWS`:** NO borra el contenido de la tabla al hacer **`COMMIT`**, sino que los borra al finalizar la sesión.
  - Son datos privados de cada sesión: No pueden compartirse.

Objetos del Esquema: VISTAS

- **VISTAS (*views*):** Una vista es una representación particular de los datos contenidos en una o más tablas (u otras vistas).
  - A diferencia de las tablas, las vistas no ocupan ningún espacio de almacena-miento. Oracle almacena la definición de las vistas en el diccionario de datos de donde la toma cuando se ejecuta una instrucción que referencia a la vista.
  - **Utilidades de las Vistas:**
    - **Seguridad:** Restringir el acceso a ciertas filas/columnas de una tabla.
    - **Esconder la complejidad de los datos:** Una vista puede incluir una operación de reunión, muchas tablas...
      - Simplificar sentencias al usuario: Evitan tener que conocer el nombre de todas las tablas y simplifican consultas habituales complejas.
    - **Presentar los datos desde otra perspectiva:** Cambiando los nombres de atributos, introduciendo operaciones...
    - **Que las aplicaciones sean independientes a cambios en las tablas base.**
    - **Efectuar consultas que no se pueden hacer sin vistas:** Como hacer una reunión entre una tabla y una consulta con **`GROUP BY`** (o con una **`UNION`**).

Objetos del Esquema: SNAPSHOTS

- **VISTAS MATERIALIZADAS (m*aterialized views*, s*napshots*):**
  - **Son útiles para:**
    - Calcular y almacenar agregaciones (sumas, medias...), reuniones o, en síntesis, consultas lentas.
    - En entornos distribuidos, pueden usarse para duplicar los datos localmente, evitando accesos lejanos y lentos.
  - **Pueden refrescarse** manualmente, a intervalos regulares de tiempo o cuando termina una transacción sobre las tablas base (*master tables*).
  - Para crearlo se debe tener los permisos para **`CREATE MATERIALIZED VIEW`** y **`CREATE TABLE`**, a parte de los permisos de acceso a las tablas base y de espacio suficiente en el *tablespace* por defecto o en el que se especifique explícitamente en la sentencia de creación.
  - **Formato:**
    ```
    CREATE MATERIALIZED VIEW <Nombre> <Cláusulas>
      AS <subconsulta>;
    ```
    - **<Cláusulas>** incluye multitud de características, entre las que destacan:
      - **`BUILD [INMEDIATE | DEFERRED]:`** Indica que la vista materializada será llenada de forma inmediata (por defecto), o lo será en la primera operación de **`REFRESH`** (refresco), el cual será un refresco completo. Hasta entonces no podrá ser usada.

Objetos del Esquema: SNAPSHOTS

- **[REFRESH <Opciones> | NEVER REFRESH]:** Establece las opciones para refrescar automáticamente o no los datos. Estas opciones pueden ser:
  - **Opciones del Modo de Refresco:**
    - **FAST:** El refresco rápido se lleva a cabo usando los cambios hechos sobre las tablas base. Esos cambios se almacenan en una tabla especial asociada a cada tabla base que se crea con:

```
CREATE MATERIALIZED VIEW LOG ON <Tabla>
INCLUDING NEW VALUES;
```

      - » Este **M.V. LOG** almacena los cambios efectuados sobre una tabla y se actualiza con cada comando DML sobre esa tabla.
      - » Un mismo **M.V. LOG** sobre cierta tabla puede servir para actualizar todos las **M.V.** que usen esa tabla.
      - » No todas las **subconsultas** pueden beneficiarse de este tipo de refresco.
      - » **INCLUDING NEW VALUES:** Especifica que en la tabla del **LOG** se incluyan los viejos y los nuevos valores. La opción por defecto es **EXCLUDING NEW VALUES**, que no sirve para hacer refrescos de tipo **FAST**.
    - **COMPLETE:** El refresco completo consiste en rehacer la consulta.
    - **FORCE:** Cuando haya que refrescar, se ejecutará un refresco **FAST** si es posible. En otro caso se hara un refresco **COMPLETE**.
      Esta es la opción por defecto.

Objetos del Esquema: SNAPSHOTS

  - **Opciones de Cuando Refrescar:**
    - **ON [COMMIT | DEMAND]:** Especifica si el refresco se ejecutará al final de cada transacción que modifique una tabla base, o bajo petición explícita (ejecutando un procedimiento del paquete **DBMS_MVIEW**).
    - **START WITH <Fecha>:** Especifica la fecha del primer refresco.
    - **NEXT <Fecha>:** Fecha para calcular el intervalo entre refrescos automáticos.
  - **FOR UPDATE:** Permite que la **M.V.** sea modificada y estos cambios pueden ser propagados a la tabla base (si se usa *Advanced Replication*).
- **Ejemplos:**
  - Usa dos tablas del esquema **Ventas** en una BD remota:

```
CREATE MATERIALIZED VIEW Ventas.Clientes_Recientes
  AS SELECT * FROM Ventas.Clientes@dbs1.uma.es C
     WHERE EXISTS (SELECT *
                   FROM Ventas.Ordenes@dbs1.uma.es O
                   WHERE C.DNI = O.DNI_Cliente);
```

  - Cada 7 días, a partir de hoy obtengo la unión de 2 tablas de sendos usuarios en BD remotas:

```
CREATE MATERIALIZED VIEW sales_emp
  TABLESPACE Mi_Tablespace
  REFRESH FAST START WITH SYSDATE NEXT SYSDATE + 7
  AS SELECT * FROM Patricia.Cosas@Granada UNION
     SELECT * FROM Miriam.Cosas@Malaga;
```

Objetos del Esquema: SECUENCIAS

- **SECUENCIAS (*Sequence*):** Genera una serie de números.
  - Útil para generar llaves primarias. En entornos multiusuario, aceleran las transacciones (no dependen unos usuarios de que terminen otros usuarios su asignación).
  - **Ej.:** `CREATE SEQUENCE NumVenta START WITH 10 INCREMENT BY 2 MAXVALUE 10000 MINVALUE 0 CYCLE ORDER;`
    - Genera los números: 10, 12, 14...
    - **`START WITH`:** Primer valor de la secuencia. El valor por defecto es el valor máximo en secuencias descendentes y el mínimo en secuencias ascendentes.
      - Admite números de hasta 28 dígitos.
    - **`INCREMENT`:** Incremento entre dos números consecutivos. Por defecto 1. Puede ser positivo o negativo, pero no cero.
    - **`MINVALUE`:** Menor valor de la secuencia. Debe ser menor que los valores de **`START WITH`** y **`MAXVALUE`**. Puede sustituirse por **`NOMINVALUE`** (opción por defecto) que especifica el valor 1 y -10^{26} para secuencias ascendentes y descendentes respectivamente.
    - **`MAXVALUE`:** Mayor valor. Debe ser mayor que los valores de **`START WITH`** y **`MINVALUE`**. Puede sustituirse por **`NOMAXVALUE`** (opción por defecto) que especifica el valor –1 y 10^{27} para secuencias ascendentes y descendentes respectivamente.
    - **`CYCLE`:** La secuencia continua tras alcanzar su valor máximo o mínimo siguiendo por el otro extremo (según sea ascendente o descendente). Su valor por defecto es **`NOCYCLE`**, que hace que la secuencia no generará más valores.
    - **`ORDER`:** Los números se generarán en orden. La opción por defecto es **`NOORDER`**, aunque se generarán por defecto si Oracle usa el modo exclusivo (no el modo paralelo).
      - Esto es útil si se quiere usar la secuencia para ordenar temporalmente las tuplas.

Objetos del Esquema: SECUENCIAS

- **Utilizar SECUENCIAS:**
  - Se usan dos pseudocolumnas cualificadas con el nombre de la secuencia:
    - **Secuencia.`CURRVAL`:** Devuelve el valor actual de la secuencia.
    - **Secuencia.`NEXTVAL`:** Incrementa la secuencia y devuelve ese valor.
      - Se incrementa independientemente de que la transacción se confirme (**`COMMIT`**) o se deshaga (**`ROLLBACK`**): El incremento no se deshace.
      - Si dos usuarios están usando la misma secuencia, cada uno de ellos obtendrá valores diferentes y puede ser que vean saltos en la secuencia.
  - Se debe tener el permiso **`SELECT`** sobre la secuencia, o el privilegio del sistema **`SELECT ANY SEQUENCE`**.
  - Si la secuencia es de otro usuario (o esquema) se debe cualificar también con ese nombre: **`Esquema.Secuencia.CURRVAL`**
  - Si en una misma sentencia se usan ambas pseudocolumnas de la misma secuencia, primero se incrementa la secuencia y, por tanto, se devuelve el mismo valor, independientemente del orden en el que estuvieran.
  - Se puede usar una misma secuencia para varias tablas o propósitos.
  - Ejemplos:
    - `SELECT NumVenta.CURRVAL FROM DUAL;`
    - `INSERT INTO Ventas(IDVenta,Producto,Fecha) VALUES (NumVenta.NEXTVAL,'Bicicleta',SYSDATE);`

Objetos del Esquema: SINÓNIMOS

- **SINÓNIMOS:** Un sinónimo es un nombre alternativo, o alias, de una tabla, vista, *snapshot*, secuencia, procedimiento, función o paquete.
  - No requieren almacenar más que su definición en el diccionario de datos.
  - Los sinónimos tienen una doble función: conveniencia y seguridad (ocultando el nombre y el propietario de un objeto, y su localización en BD distribuidas).
  - Un sinónimo público es aquel cuyo propietario es el grupo de usuarios **PUBLIC** y todos los usuarios pueden acceder a él.
  - Un sinónimo privado pertenece al subesquema de un determinado usuario que puede controlar el uso del mismo por el resto de usuarios.
  - **Formato:** `CREATE [PUBLIC] SYNONYM <Nombre> FOR <Objeto>;`
  - **Ejemplos:**
    - `CREATE SYNONIM pventa FOR Paco.Proyect_Venta;`
      - Crea un sinónimo privado para una tabla del esquema de **Paco**.
    - `CREATE PUBLIC SYNONYM Prod FOR Scott.Prod@Ventas;`
      - Crea un sinónimo público para una tabla de **Scott** en una BD remota llamada **Ventas**.
  - Si un sinónimo público para una tabla tiene el mismo nombre que una tabla de un usuario, para ese usuario no tendrá efecto el sinónimo.

Objetos del Esquema: CLUSTERS

- **CLUSTER:** Es un grupo de tablas que comparten las mismas páginas porque tienen alguna columna en común y suelen usarse juntas.
  - **Ejemplo:** Si construimos un *cluster* con las tablas **EMPLEADOS** y **DEPARTAMENTOS** utilizando la columna común **CODIGO_DEPARTAMENTO**, los empleados de un mismo departamento, junto con los datos del departamento, compartirían las mismas páginas, y cada valor clave del *cluster* se almacenará sólo una vez.
    - Si cambia el valor clave del *cluster* para una fila, Oracle la realojará.
  - **Formato de las Páginas:** En general, el de un *cluster* no difiere del resto de páginas, excepto la inclusión de datos en el directorio de la tabla.
    - Cuando creamos un *cluster* debemos especificar la cantidad media de espacio requerido para almacenar todas las filas correspondientes a un valor clave: Parámetro **SIZE** de la instrucción **CREATE CLUSTER**.
    - **SIZE** determina el máximo número de claves que pueden ser almacenadas en una página.
      - Por ejemplo, si cada página tiene 1700 bytes de espacio disponible, y el tamaño de la clave es de 500 bytes, cada página podría contener tres claves.

Objetos del Esquema: CLUSTERS

- **Es obligatorio construir un Índice del *cluster***, para las columnas de la clave del *cluster*. Es un índice definido específicamente para el *cluster*.
  - No puede ejecutarse ninguna sentencia DML en las tablas del cluster, si no existe su índice correspondiente.
  - El índice será utilizado para localizar una fila en el *cluster*, ya que apunta a la página asociada con cada valor de clave del *cluster*.
- ***Cluster HASH*:** Existe otra forma de crear un *cluster* mediante la utilización de accesos directos a las páginas donde deben ser almacenadas las filas, en vez de hacerlo según el índice definido anteriormente.
  - Crear en primer lugar un ***hash cluster*** y cargar posteriormente los datos de las tablas asociadas en él.
  - **Función *hash*:** Función que se aplica a la columna o columnas de la clave del *cluster*, para obtener la página que corresponde a las filas de esa clave.
  - Esto ahorra tener que leer el índice para localizar o insertar un dato, ya que la función *hash* NO requiere lecturas de disco.
  - El parámetro **HASHKEYS** de la sentencia **CREATE CLUSTER** limita el número de valores *hash* que genera la función. Oracle redondea ese número al primo más cercano (por ejemplo, si se especifica 100, Oracle usará 101):
    - Si **HASHKEYS** es grande se reduce la probabilidad de colisión.

Objetos del Esquema: ÍNDICES

- **ÍNDICES:** Estructura opcional asociada con una tabla o un *cluster*.
  - Se crean sobre **una o varias columnas**, para acelerar la ejecución de sentencias SQL.
    - Tras cada columna puede especificarse **ASC** o **DESC**.
  - Una tabla puede tener cualquier **cantidad de índices**, si la combinación de columnas en cada uno sea diferente. Incluso, cambiando el orden:
    - Ejemplos:

```
CREATE INDEX Pieza_idx1 ON Pieza (Nombre, Cantidad);
CREATE INDEX Pieza_idx2 ON Pieza (Cantidad, Nombre);
```

  - Podemos utilizar distintos **Tipos de Índices** que permiten una funcionalidad distinta de cara al rendimiento de la BD**:** Árboles B sobre las tablas, árboles B sobre los clusters, índices *hash* sobre clusters, índices de clave inversa e índices de mapas de bits.
  - Los índices son lógica y físicamente **Independientes de los Datos** de las tablas asociadas y son mantenidos dinámica y automáticamente.
    - Se puede crear o borrar un índice sin ningún efecto lateral sobre los datos de la tabla u otros índices.
  - **Operaciones cuando existen índices:**
    - **Consultar:** El tiempo de acceso es casi constante, aunque se inserten más filas en la tabla.
    - **Borrar, Insertar o Actualizar:** El tiempo aumenta si existen muchos índices sobre la tabla (Oracle debe también actualizar los índices).

Objetos del Esquema: ÍNDICES

- Pueden ser **únicos** (*unique*) o **no únicos** (*nonunique*), según exijan o no que las columnas del índice admitan o no valores dublicados en distintas filas: **`CREATE [UNIQUE] INDEX...`**
  - Si esa restricción existe en la tabla (**`UNIQUE`**), Oracle crea un índice único automáticamente.
    - Oracle no recomienda crear índices únicos explícitamente.
- Es recomendable que sean **únicos** y que, al menos, se cree **uno por cada clave primaria o externa** de cada tabla, así como por cada columna que contenga valores de búsqueda usuales, sin excedernos.
- **Índices Basados en Función (*function-based indexes*):** Se puede construir un índice sobre una función determinística definida por el usuario.
  - **Ej. 1:** Índice que será usado en consultas como la propuesta:
    - **`CREATE INDEX idx ON table_1 (a + b * (c - 1), a, b);`**
    - **`SELECT a FROM table_1 WHERE a + b * (c - 1) < 100;`**
  - **Ej. 2:** Para facilitar búsquedas insensibles a mayúsculas/minúsculas:
    - **`CREATE INDEX uppercase_idx ON Pieza (UPPER(Nombre));`**
    - **`SELECT * FROM Pieza WHERE UPPER(Nombre) = 'TORNILLO';`**

Objetos del Esquema: ÍNDICES

- **Cuando se Crea un Índice:**
  - Se le asigna un segmento de índice para contener sus valores en el *tablespace* correspondiente.
    - Es preferible que este *tablespace* no sea el mismo en el que está contenida la tabla asociada y que ambos *tablespaces* estén almacenados en discos diferentes, para que Oracle pueda leerlos en paralelo.
  - Al crear un índice, Oracle ordena las columnas del índice y almacena el valor de los índices junto con el **`ROWID`** de las filas.
  - Los índices pueden crearse en orden ascendente (**`ASC`**), descendente (**`DESC`**), comprimidos (**`COMPRESS`**) o no comprimidos (**`NOCOMPRESS`**).
- **Indices de Clave Inversa:** A veces, puede ocurrir que las inserciones o modificaciones a un índice se concentren en un conjunto pequeño de bloques. Esto puede disminuir considerablemente el rendimiento debido a los continuos bloqueos de los mismos bloques del índice.
  - En estos casos, podemos generar un índice inverso, que consiste en que las claves (valores de las columnas) se insertan invirtiendo el orden de los bytes: Por ejemplo ABEL se almacenaría como LEBA.
  - De esta forma los valores adyacentes quedan diseminados por el índice.
  - Esta posibilidad es recomendable tan solo para operaciones de acceso a un único valor, ya que las recuperaciones por rango de índice no se beneficiarán del índice al no estar ahora las entradas ordenadas alfabéticamente por el valor de la columna.
  - Se crean añadiendo **`REVERSE`** al final de la instrucción de creación:

    `CREATE INDEX i ON t (a, b, c) REVERSE;`
  - Convertir un índice invertido en uno normal: **`ALTER INDEX i REBUILD NOREVERSE;`**
    - El proceso inverso no puede hacerse: Habría que crearlo de nuevo.

Objetos del Esquema: ÍNDICES

El **Propósito de un Índice** es mantener punteros a las filas de una tabla que contiene los valores de la clave de indexación. En un índice estándar, esto se consigue almacenando el valor los **ROWID** de las filas de cada clave.

– **Indices de Mapas de Bits** (**CREATE BITMAP INDEX**):
  - En un **índice de mapa de bits**, para cada valor de clave se utiliza un **mapa de bits en vez de el ROWID.**
  - Cada **bit del mapa de bits** corresponde a un **ROWID**, y vale **1** si la fila pertenece a ese valor clave, y **0** si no pertenece.
    – Una función de mapeo convierte la posición del bit en un **ROWID**, de manera que la funcionalidad de este índice es la misma que la de un índice estándar pero utilizando una diferente representación interna.

Ejemplo: - TABLA:

CODIGO	E_CIVIL	...	PROVINCIA
101	Soltero	...	Málaga
102	Casado	...	Madrid
103	Soltero	...	Barcelona
104	Divorciado	...	Barcelona
105	Soltero	...	Madrid
106	Casado	...	Madrid

- ÍNDICE:

VALOR	MAPA DE BITS
Barcelona	0 0 1 1 0 0
Madrid	0 1 0 0 1 1
Málaga	1 0 0 0 0 0

  - Si el número de valores distintos de clave es pequeño (esto es, la cardinalidad de la columna es baja), los índices de mapas de bits son muy eficientes desde el punto de vista del ahorro de espacio: Un índice **no** puede ser **BITMAP** y **UNIQUE** a la vez.
    – En general puede decirse que las columnas cuyos valores se repiten más de cien veces, son buenas candidatas para un índice de mapas de bits.
  - Para las columnas que tienen valores de clave con un bajo índice de repetición, esta forma de indexación también se muestra muy eficiente en casos en los que la definición del índice responde a **numerosas condiciones** de una cláusula **WHERE**, ya que las filas que satisfacen algunas, pero no todas las condiciones, son filtradas y desconsideradas antes de que la tabla sea accedida.
    – Esto puede provocar una mejora sustancial en el tiempo de realización de la consulta.

Objetos del Esquema: Tablas Org. Ind.

- **Tablas Organizadas por Índice** (*index-organized tables*)**:** Son tablas en las que los datos están contenidos en el índice asociado (*B-tree*).
  – Con la sentencia **CREATE TABLE** y su cláusula **ORGANIZATION INDEX**.
  – Las **modificaciones** en los datos de la tabla (insertar, actualizar o borrar) tienen como efecto una actualización del índice.
  – Realmente, **el índice es la tabla:** En vez de estar compuesto de valor de clave y puntero (**ROWID**), se compone de **valor de clave y resto de valores de la fila**.
    - No duplica el almacenamiento de las claves como en una tabla ordinaria con su índice.
  – Son idóneas para **accesos por clave primaria** pero no recomendadas para otro tipo de accesos.
  – Pueden crearse **índices adicionales** sobre este tipo de tablas para acceder eficientemente por otras columnas.
  – **Diferencias principales** entre estas tablas y las tablas ordinarias**:**

Tabla Ordinaria	Tablas Organizadas por Índice
ROWID identifica una fila.	La llave primaria identifica una fila.
Llave primaria opcional.	Llave primaria obligatoria.
Acceso por el **ROWID**.	Acceso por la llave primaria.
Análisis secuencial para recuperar todas las filas.	Un análisis completo del índice recupera todas las filas ordenadas por la PK.
Pueden almacenarse en un cluster.	No pueden almacenarse en un cluster.
Pueden contener columnas de tipos **LONG** y LOB.	Pueden contener columnas LOB, pero no **LONG**.

Control de Concurrencia/Consistencia

- Una BD puede sufrir múltiples **Accesos Simultáneos** a los mismos datos: El control de la ejecución simultánea de las transacciones para asegurar la **Consistencia** (estado coherente) de los datos es vital.
  - Mientras que el **Aislamiento entre las Transacciones** es generalmente deseable, el funcionamiento de muchas aplicaciones en este modo puede comprometer seriamente rendimiento.
  - El estándar ANSI/ISO **SQL 92** define **Cuatro Niveles de Aislamiento** de una transacción con diferentes grados de impacto sobre el proceso de transacciones. Estos niveles de aislamiento se definen en relación a **Tres Fenómenos** que deben prevenirse cuando se produce concurrencia de transacciones. Los tres fenómenos referidos son:
    - **Lecturas Erróneas** (*dirty reads*)**:** Una transacción lee datos que han sido escritos por otra transacción que aún no ha sido confirmada (con **`COMMIT`**).
    - **Doble Lectura** (*non-repeatable reads*)**:** Una transacción lee datos que ya había leído, encontrándose que, entre las dos lecturas, los datos han sido modificados o borrados por una transacción que ya ha sido confirmada.
    - **Lectura Fantasma** (*phantom read*)**:** Una transacción reejecuta una consulta encontrando que el conjunto de filas resultantes ha sido ampliado por otra transacción que insertó nuevas filas y que ya ha realizado su **`COMMIT`**.

Control de Concurrencia/Consistencia

- Los **4 Niveles de Aislamiento de SQL 92** son los siguientes:

Nivel de Aislamiento	Lectura Errónea	Doble Lectura	Lectura Fantasma
Read Uncommited	Posible	Posible	Posible
Read Commited	No posible	Posible	Posible
Repeatable Read	No posible	No posible	Posible
Serializable	No posible	No posible	No posible

- **Oracle ofrece estos Niveles de Aislamiento:**
  - **`READ COMMITED`:** Cada consulta que ejecuta una transacción ve solamente los datos que han sido confirmados con anterioridad al inicio de ejecución de la consulta (no de la transacción): **Consistencia a Nivel de Sentencia**.
    - De esta forma, nunca se leerán datos sin confirmar.
    - Es el nivel de aislamiento por defecto.
    - Si una sentencia DML requiere modificar filas que están bloqueadas por otra transacción, la sentencia esperará hasta que se libere el bloqueo.
    - Es equivalente a **`READ WRITE`**.
  - **`SERIALIZABLE`:** Las transacciones ven solamente los datos que han sido confirmados con anterioridad al inicio de la transacción, exceptuando los cambios que ellas mismas realicen: **Consistencia a Nivel de Transacción**.
  - **`READ ONLY`:** Igual que el **`SERIALIZABLE`**, pero además no permite que la transacción realice ninguna modificación en los datos. Este nivel no es de SQL 92.
    - Los datos que ve la transacción actual son los datos confirmados (*committed*) antes de que la transacción empezara.
    - Esto es útil para informes que consultan muchas tablas que están continuamente siendo actualizadas.

Control de Concurrencia/Consistencia

- **Puede cambiarse de Nivel:**
  - Al **Principio de una Transacción**, con la orden**:**

```
SET TRANSACTION
  ISOLATION LEVEL {SERIALIZABLE | READ COMMITTED};
```

    o bien:

```
SET TRANSACTION {READ ONLY | READ WRITE};
```

  - Para todas las transacciones siguientes en una **Sesión:**

```
ALTER SESSION SET
  ISOLATION_LEVEL = {SERIALIZABLE | READ COMMITTED};
```

- **Puede cambiarse de segmento de *Rollback* en una Transacción:**

```
SET TRANSACTION USE ROLLBACK SEGMENT <nombre_seg_Rollback>;
```

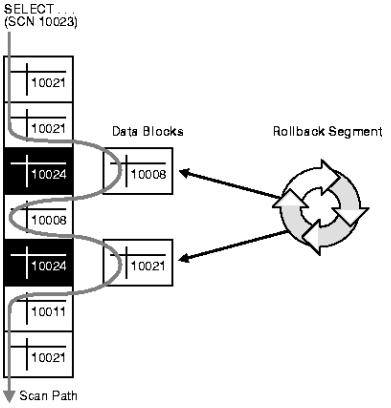
  - Implicitamente establece esta transacción como **READ WRITE**, ya que el modo **READ ONLY** no genera información en los segmentos de *rollback*.
  - Es útil en transacciones que modifican muchos datos, para las que posiblemente no haya espacio en el segmento de *rollback* por defecto.
- La sentencia **SET TRANSACTION** sólo puede usarse al principio de una transacción.

Control de Concurrencia/Consistencia

- **Consistencia de un ambiente Multiusuario:** Oracle mantiene la Consistencia mediante el uso de un modelo de consistencia multiversión y varios tipos de bloqueos y transacciones.
- **Control de la Concurrencia Multiversión:**
  - **Consistencias a Nivel de Lecturas:** Que todos los datos leidos provengan de un único momento temporal, aunque otra transacción termine durante su ejecución. Observe que esto se aplica a la sentencia **SELECT**, pero también a **UPDATE**, **DELETE** y también a **INSERT** con una subconsulta.
    - **Consistencia a Nivel de Sentencia** (*statement-level read consistency*)**:** De forma automática se permite lectura coherente, o consistente, de los datos resultantes de una consulta.
    - **Consistencia a Nivel de Transacciones** (*transaction-level read consistency*)**:** Consistencia para todas las consultas incluidas en una transacción.
      - Por supuesto, una transacción siempre ve los cambios hechos anteriormente por ella misma.
      - Este nivel evita la lectura fantasma y la lectura doble.
- La **Consistencia a Nivel de Sentencia** siempre está garantizada por Oracle. En cambio, para garantizarla a **Nivel de Transacción** debemos utilizar el **Nivel de Aislamiento SERIALIZABLE**.

Control de Concurrencia/Consistencia

- Para conseguir la Consistencia a Nivel de Sentencia o de Transacciones, Oracle utiliza la información incluida en los **Segmentos de *Rollback***.
  - **Oracle** asigna a cada transacción un número **SCN** (*System Change Number*) que graba junto con los datos en el segmento de rollback.
  - Los datos cambiados más recientemente (SCN más reciente que la consulta o transacción en curso) son reconstruidos usando los datos de los segmentos de *rollback*.
  - En la **Figura**, la consulta en ejecución tiene el **SCN** 10023, por lo que esa consulta sólo leerá los datos cambiados con anterioridad.
  - Los datos con SCN mayor a 10023 son leídos de los segmentos de *rollback*.



Control de Concurrencia/Consistencia

- En **Consultas de Larga Ejecución** (*long-running queries*), puede ocurrir (pero es raro), que **Oracle no pueda devolver unos resultados coherentes**, reconstruyendo la información con los segmentos de *rollback*, porque esos datos ya no estén allí por haber sido reescritos a causa de la forma circular del segmento de *rollback*.
  - **Esto ocurrirá con mayor probabilidad:**
    1. Si los **segmentos de *rollback*** son demasiado **pequeños** y/o
    2. Si el **nivel de modificaciones** de la BD es **muy alto**.
  - En estos casos se produce un mensaje de **error** informando al usuario de este hecho: **`ORA-1555: snapshot too old (rollback segment too small)`**
  - **Soluciones:**
    - Aumentar el tamaño de los segmentos de rollback o crear nuevos.
    - Ejecutar esas operaciones cuando haya pocas transacciones en curso.
    - Obtener un bloqueo compartido (**SHARE**) sobre la tabla de la consulta, para evitar bloqueos de otros usuarios durante la transacción.
- Si tenemos **Consistencia a Nivel de Transacciones**, esto también puede ocurrir, con mayor probabilidad, en **Transacciones Largas**.

Bloqueos DML (DML Locks)

- **BLOQUEOS (*locks*):** Mecanismos para prevenir los problemas de interacción entre usuarios accediendo al mismo recurso.
  - **Dos niveles de Bloqueo en una BD multiusuario:**
    - **Bloqueos Exclusivos (*exclusive locks*):** No permite compartir recursos. La primera transacción que bloquea un recurso exclusivamente es la única que puede usarlo hasta que lo libere.
    - **Bloqueos Compartidos (*share locks*):** Permite compartir recursos dependiendo del tipo de operaciones involucradas.
      - Varios usuarios "leyendo" pueden compartir datos, pero no un usuario "escribiendo" (que usará un bloqueo exclusivo).
  - **Dos Tipos de Bloqueos:**
    - **Bloqueo de Fila (TX):** Este es el menor bloqueo posible.
      - Se produce cuando se ejecuta una instrucción **`INSERT`**, **`UPDATE`**, **`DELETE`** o **`SELECT`** con la cláusula **`FOR UPDATE`**.
      - No permite la lectura ni la modificación de la fila por otra transacción.
      - Cuando una transacción obtiene este tipo de bloqueo exclusivo de una fila, también obtiene un **bloqueo de tabla** en la correspondiente tabla.
    - **Bloqueo de Tabla (TM):**
      - Una transacción obtiene este tipo de bloqueo de toda la tabla cuando realiza una operación **DML** de las enumeradas anteriormente o usa la sentencia **`LOCK TABLE`**.
      - Evita que otra transacción consiga un **bloqueo DDL**, el cual es necesario para efectuar alguna operación de **DDL** sobre la tabla.

Bloqueos DML (DML Locks)

- Un **Bloqueo de una Tabla** se puede producir de varios **MODOS** que determinan las operaciones que se permiten a las otras transacciones sobre la misma tabla**:**
  - ***RS*: `ROW SHARE` *Table Lock:***
    - Indica que la transacción ha bloqueado filas de la tabla y va a modificarlas.
    - Permite compartir la tabla con otras transacciones en cualquier modo salvo que soliciten un bloqueo de las mismas filas, o bien, un bloqueo exclusivo de la tabla.
    - Sentencias SQL que bloquean RS:
      ```
      SELECT ... FROM ... FOR UPDATE OF ... ;
      LOCK TABLE <Tabla> IN ROW SHARE MODE;
      ```
  - ***RX*: `ROW EXCLUSIVE` *Table Lock:***
    - Indica que la transacción ha realizado bloqueos sobre algunas filas de la tabla. Se produce con las instrucciones **`INSERT`**, **`UPDATE`**, **`DELETE`**.
    - El resto de transacciones no puede bloquear la tabla con ninguno de los modos siguientes.
  - ***S*: `SHARE` *Table Lock:***
    - Se adquiere cuando se ejecuta la instrucción **`LOCK TABLE`** asociada.
    - No permite realizar modificaciones en la tabla, aunque sí lecturas (con y sin **`FOR UPDATE`**), así como otros bloqueos del mismo tipo S, o de tipo RS.
    - Si varias transacciones tienen este bloqueo para la misma tabla, ninguna podrá modificar la tabla, aunque haya ejecutado la sentencia **`SELECT...FOR UPDATE`**.
  - ***SRX*: `SHARE ROW EXCLUSIVE` *Table Lock:***
    - Se adquiere mediante la instrucción **`LOCK TABLE`** asociada y, en un momento dado, sólo puede conseguir este tipo de bloqueo una transacción para una tabla concreta.
    - Solamente permite a otras transacciones realizar lecturas de la tabla o bloquear ciertas filas con la sentencia **`SELECT...FOR UPDATE`**, pero no permite modificar la tabla.
  - ***X*: `EXCLUSIVE` *Table Lock:***
    - Es el modo más restrictivo de bloqueo y permite, a la transacción que lo obtiene, un acceso en exclusiva para escribir en la tabla.
    - Sólo permite a otras transacciones consultas normales.

Bloqueos DML (DML Locks)

– **Resumen de los MODOS de Bloqueo de una Tabla:**
  - Operaciones que producen cada modo.
  - Modos en los que se permite cada tipo de operación en otra transacción:
    – Operaciones permitidas o no cuando otra transacción tiene cierto modo de bloqueo.

Instrucción SQL	Bloqueo de Fila	Bloqueo de Tabla	¿Modos Permitidos? RS	RX	S	SRX	X
`SELECT ... FROM table`			Sí	Sí	Sí	Sí	Sí
`INSERT INTO table ...`	TX	RX	Sí	Sí	No	No	No
`UPDATE table ...`	TX	RX	Sí*	Sí*	No	No	No
`DELETE FROM table ...`	TX	RX	Sí*	Sí*	No	No	No
`SELECT...FROM table ...FOR UPDATE OF...`	TX	RS	Sí*	Sí*	Sí*	Sí*	No
`LOCK TABLE table IN ...`							
`ROW SHARE MODE`		RS	Sí	Sí	Sí	Sí	No
`ROW EXCLUSIVE MODE`		RX	Sí	Sí	No	No	No
`SHARE MODE`		S	Sí	No	Sí	No	No
`SHARE ROW EXCLUSIVE MODE`		SRX	Sí	No	No	No	No
`EXCLUSIVE MODE`		X	No	No	No	No	No

* Sí, si no existe otra transacción con bloqueos de fila conflictivos; En otro caso se produce una espera.

– **Sentencia `LOCK TABLE`:** Bloquea una tabla o vista en el modo indicado.

```
LOCK TABLE <Tabla> IN <Modo> MODE [NOWAIT];
```

  - **`NOWAIT`:** Indica que si no puede hacer el bloqueo, que no espere y produzca un error indicando que la tabla está bloqueada por otro usuario.

Deadlocks (Muertes por Bloqueos)

- **Un Deadlock ocurre cuando dos o más usuarios están esperando, respectivamente, datos que están bloqueados por el otro usuario.**
  – **Ejemplo:**

Usuario 1	Usuario2
`UPDATE Pieza`	`UPDATE Pieza`
`SET Nombre='TUERCA'`	`SET Peso=10`
`WHERE P# = 1000;`	`WHERE P# = 2000;`
`UPDATE Pieza`	`UPDATE Pieza`
`SET Precio=Precio*1.1`	`SET Peso=10`
`WHERE P# = 2000;`	`WHERE P# = 1000;`
`ORA-00060: Deadlock detected while waiting for resource.`	

Sin problemas: Cada transacción tiene un bloqueo a distinta fila.

Problema: Ninguna transacción puede conseguir el recurso que necesita.

***Deadlock*:** Oracle informa del problema a una transacción y la deshace (*rollback*).

  – **Los deadlocks son raros en Oracle**, ya que sus bloqueos son por cada fila y no por páginas.
    - Además, Oracle sólo bloquea por defecto las filas necesarias y no bloquea una tabla completamente aunque muchas de sus filas estén bloqueadas (*lock escalation*).
  – **Los *Deadlocks* entre varias tablas pueden evitarse si las transacciones que acceden a esas tablas las bloquean en el mismo orden, a través de bloqueos implícitos o explícitos.**

Administración del SGBD Oracle

- **Funciones del Administrador o DBA:**
  - **Instalación y actualización del *software* de Oracle** (servidor y aplicaciones).
    - Cambiar claves iniciales de las 2 cuentas DBA que Oracle crea automáticamente al crear una BD**:**
      - **`SYS/CHANGE_ON_INSTALL:`** Todas las tablas y vistas del diccionario de datos pertenecen al esquema de **SYS**, y nadie debería modificarlas. Tampoco se deben crear nuevas tablas de la BD en las cuentas del DBA.
      - **`SYSTEM/MANAGER:`** Crea nuevas tablas y vistas con información administrativa.
  - **Evaluación del *hardware*:** Evaluar discos, memoria... asignar espacios de almacenamiento y planificar requerimientos futuros.
  - Planificación de los **parámetros de creación** de la BD.
  - **Creación de la BD:**
    - Estructuras de almacenamiento (*tablespaces*...) .
    - Implementación del diseño de la BD: Objetos (tablas, vistas...) y restricciones.
  - Modificar la **estructura de la BD** cuando sea necesario.
  - **Apertura y cierre** de la BD.
  - **Gestión de usuarios** y Sistemas de **seguridad:** Permisos y Roles.
  - **Auditoría:** Controlar y monitorizar el acceso a la BD.
  - **Copias de seguridad** (*backup*) y sus recuperaciones (*recovery*).
  - **Afinamiento** de la BD (optimizar su rentimiento).

Administración del SGBD Oracle

- **Utilidades de Administración:**
  - **SQL*Loader:** Se usa para cargar datos desde ficheros del S.O. a tablas de la base de datos.
    - Puede usarse por usuarios y administradores.
    - Permite especificar el formato de entrada de los datos.
  - **Export e Import:** Permiten mover datos entre distintas BD Oracle.
    - **Export** guarda los datos en ficheros, e **Import** lee esos ficheros y carga los datos en las tablas: Pueden usarse como medios para *backup*.
    - Puede ser entre versiones de Oracle de distintos S.O.
- Ver las **Versiones de los distintos productos Oracle** (núcleo, PL/SQL...): Se puede hacer consultando la vista del diccionario de datos llamada**:** **`PRODUCT_COMPONENT_VERSION`**.
- Existen **Dos Privilegios Importantes** para la Administración (no son roles)**:**
  - **SYSOPER:** Puede hacer casi todas las tareas de administración, excepto conceder la administración a otros, crear una BD y poco más.
  - **SYSDBA:** Contiene todos los privilegios del sistema **`WITH ADMIN OPTION`** (incluyendo **SYSOPER**) y permite usar **`CREATE DATABASE`**.
  - Se **conceden** normalmente **`: GRANT SYSDBA TO scott;`**
  - Y se **revocan** de similar forma **`: REVOKE SYSDBA FROM scott;`**
  - Usuario se puede **conectar** con**`: CONNECT scott/tiger AS SYSDBA`**
    - Se conecta al esquema por defecto (**`PUBLIC`** y **SYS** respectivamente), y no al esquema asociado al usuario, por lo que éste no podrá ver sus tablas sin cualificarlas con su nombre de usuario.

Crear una BD Oracle

- **Creación de la Base de Datos:** Pasos para crear una BD Oracle:
  - **Realizar Copias de Seguridad de otras posibles BD preexistentes.**
  - **Crear el Fichero de Parámetros:** Las instancias se arrancan a partir de un fichero de parámetros.
    - Para cada base de datos, debe tenerse un fichero de parámetros de este tipo.
      - Oracle suministra un fichero de parámetros de inicialización por defecto que puede ser editado y modificado para cada base de datos.
    - **Los parámetros son:**
      - **DB_NAME:** Nombre local de la base de datos que no puede ser cambiado.
      - **DB_DOMAIN:** Localización lógica de la BD en la red (por ejemplo SCI.UMA.ES). En combinación con **DB_NAME** debe identificar un único nombre en la red.
      - **CONTROL_FILES:** Si no especifica nada, Oracle creará uno por defecto. Se recomienda tener al menos dos ficheros de control en distintos discos.
      - **DB_BLOCK_SIZE:** Es el tamaño de página de la BD. Por defecto es el mismo tamaño que el de un bloque del S.O. (2048 ó 4096 bytes).
      - **DB_BLOCK_BUFFERS:** Número de *buffers* en la caché del SGA. Normalmente tiene un valor mínimo de 1000 ó 2000.
      - **PROCESSES:** Máximo número de procesos que pueden conectarse a la BD simultáneamente. Debe incluir los 5 procesos de *background* más uno por cada usuario. Si se estiman 50 usuarios concurrentes como máximo, este valor puede ser de 55.
      - **ROLLBACK_SEGMENTS:** Es una lista de los segmentos de *rollback* que la Instancia asigna cuando arranca la BD.

Crear una BD Oracle

- **Arrancar la Instancia:** Mediante la instrucción **STARTUP** de SQL*Plus
  - Puede existir un programa que haga esta parte o todo el proceso, pero si se hace manualmente, en este paso debe arrancarse la instancia (SGA, procesos de *background*...) con la opción **NOMOUNT**.
- **Crear la BD:** Se hace mediante la instrucción **CREATE DATABASE**, lo que provoca que se realicen las siguientes operaciones automáticamente:
  - Crea los ficheros de datos para la BD (*datafiles*)**: ALTER TABLESPACE**
  - Crea los ficheros de control (*control files*)**: CREATE CONTROL FILE**
  - Crea los dietarios o registros de rehacer (*redo log*).
  - Crea el *tablespace* SYSTEM y el segmento de *rollback* SYSTEM**: CREATE TABLESPACE**
  - Crea el diccionario de datos.
  - Crea a los usuarios **SYS** y **SYSTEM**.
  - Especifica el conjunto de caracteres que se almacenarán.
  - Monta y abre la BD para su uso.
- **Realizar una Copia de Seguridad de la BD.**

*Apertura y Cierre, con SQL*Plus*

- **APERTURA de la BD:**
  - **Con la sentencia `STARTUP`** (que arranca la instancia).

```
STARTUP [PFILE=filename] [EXCLUSIVE] [PARALLEL]
        [MOUNT [dbname] | OPEN [open_options] [dbname] | NOMOUNT]
```

    - **`PFILE:`** Especifica el fichero de parámetros.
    - **`EXCLUSIVE:`** La instancia se asociará a la BD en exclusiva y no permite otras instancias.
    - **`PARALLEL:`** Si se van a usar varias instancias para acceder a la BD.
    - **`MOUNT:`** Monta la BD con el nombre **`dbname`**, pero no la abre. Si no se especifica nombre lo toma del parámetro de inicialización **`DB_NAME`**.
    - **`OPEN:`** Monta y abre la BD especificada con las opciones **`open_options`**:
      `READ {ONLY | WRITE [RECOVER]} | RECOVER`
    - **`NOMOUNT:`** No monta (ni abre) la BD.
  - Si se arranca sin montar la BD**: `ALTER DATABASE <nombre> MOUNT;`**
  - Si montamos la BD sin abrirla**: `ALTER DATABASE <nombre> OPEN <modo>;`**
    - donde **`<modo>`** es opcional y puede ser**:**
      - **`READ ONLY:`** No puede modificarse. No puede ser **`READ WRITE`** en otra instancia.
      - **`READ WRITE RESETLOG:`** Borra toda la información del registro de rehacer.
      - **`READ WRITE NORESETLOG:`** No borra toda esa información.
- **CIERRE de la BD:** Tres tipos:
  - **`SHUTDOWN INMEDIATE:`** Las transacciones en curso hacen *rollback* y se cierra.
  - **`SHUTDOWN [NORMAL]:`** El normal. Espera a que terminen las conexiones (usuarios) y no admite nuevas. Como el tipo anterior, no requiere hacer *recovery* en el siguiente **`STARTUP`**.
  - **`SHUTDOWN ABORT:`** El más rápido. Borra la instancia sin cerrar o desmontar la BD. Requiere hacer un *recovery* posterior.

Gestión de Usuarios

- **Crear Perfiles de Usuario:** Simplifican la gestión de usuarios. Ej.:

```
CREATE PROFILE Perfil_1 LIMIT
  SESSIONS_PER_USER 3     -- Máximo núm. de sesiones para ese usuario.
  CONNECT_TIME UNLIMITED  -- Duración máxima de la conexión en minutos.
  IDLE_TIME 10            -- Minutos de tiempo muerto en una sesión.
  FAILED_LOGIN_ATTEMPTS 4 -- nº máximo de intentos.
  ACCOUNT_LOCK_TIME 5     -- Tiempo en que queda bloqueada la cuenta.
  PASSWORD_LIFE_TIME 90   -- Nº de días de expiración de la password.
  PASSWORD_GRACE_TIME 3;  -- Periodo de gracia después de los 90 días.
```

- **Crear Usuarios: La única cláusula obligatoria es `IDENTIFIED`:**
  - Si se usa **`IDENTIFIED EXTERNALLY`,** Oracle comprueba que el nombre del usuario coincide con el nombre del usuario del S.O.
  - Las cuotas máximas se pueden especificar en Megabytes (**`M`**), en Kbytes (**`K`**), en bytes (por defecto), o indicar que no hay límite (**`UNLIMITED`**).

```
CREATE USER Pepe IDENTIFIED BY Clave_Pepe
  TEMPORARY TABLESPACE temp_ts -- Tablespace para los seg. temporales.
  DEFAULT TABLESPACE data_ts   -- Tabl. por defecto para sus objetos.
  QUOTA 100M ON test_ts        -- Máximo espacio en ese Tablespace.
  QUOTA 500K ON data_ts
  PROFILE Perfil_1;            -- Asigna el Perfil_1 al usuario.
```

  - Un usuario recién creado no tiene ningún privilegio. Se deben conceder algunos permisos. Como mínimo el rol de conexión **`CONNECT`**, aunque también es típico el de **`RESOURCE`**.
  - **Modificar un usuario: `ALTER USER`** (con similar formato).
  - **Borrar un usuario: `DROP USER Pepe CASCADE;`**
    - Si el esquema del usuario no está vacío, se debe poner **`CASCADE:`** para borrar todos sus objetos.

Auditoría de la Base de Datos

- **Auditoría:** Controla los accesos y usos de los objetos de la BD.
  - Casi todas las instrucciones SQL pueden auditarse**: ALTER**, **AUDIT**, **COMMENT**, **DELETE**, **EXECUTE**, **GRANT**, **INDEX**, **INSERT**, **LOCK**, **RENAME**, **SELECT**, **UPDATE**…
  - Inicializar el **Parámetro** en el fichero de parámetros iniciales (usualmente **INIT.ORA**): **AUDIT_TRAIL = [TRUE | FALSE]**.
- **Formato:**

```
AUDIT {Sentencias|Privilegios}
[BY <Users>] [BY SESSION | BY ACCESS]
[WHENEVER [NOT] SUCCESSFUL]
```

  - Se especifican las sentencias a controlar explícitamente o las sentencias que incluyen ciertos privilegios. Con la cláusula **ON** si es para un objeto específico.
  - Puede ser para una lista de usuarios o para todos ellos (por defecto).
  - Puede controlarse los usos por sesión o por cada vez que accede.
  - Puede controlarse sólo si las sentencias tienen éxito o sólo si no lo tienen.
  - **Ej.:**

```
AUDIT SESSION;                          AUDIT ALL;
AUDIT DELETE ANY TABLE;
AUDIT SELECT TABLE, INSERT TABLE,
      DELETE TABLE, EXECUTE PROCEDURE
      BY scott,pepe BY ACCESS;
AUDIT DELETE ON pepe.empleados;
```

- **Desactivar:** Sentencia **NOAUDIT**, con similar formato.
  - **Ej.:**

```
NOAUDIT ROLE;                           NOAUDIT ALL;
```


Auditoría de la Base de Datos

- El **Diccionario de la BD** contiene una **tabla** llamada **SYS.AUD$** que puede contener información sobre las operaciones realizadas por los usuarios.
  - Para un más fácil manejo de esta tabla, existen una serie de vistas que se crean ejecutando el *script* de SQL **CATAUDIT.SQL** que son de mucha utilidad a la hora de visualizar la información recogida.
    - Esas vistas se borran con el *script* **CATNOAUD.SQL**.
  - Dependiendo de los objetos auditados se recoge distinto tipo de información. En cualquier caso, siempre se recoge sobre el usuario, sesión, terminal, objeto accedido, operación realizada y finalización de la operación.
- Borrar información auditada siendo **SYS**:

```
DELETE FROM AUD$ WHERE obj$name='EMPLEADOS';
```

- Ver parámetros de la auditoría actual respecto a las sentencias (*statement*):

```
SELECT * FROM DBA_STMT_AUDIT_OPTS;
```

- Ver opciones de la auditoría actual respecto a los privilegios (*privileges*):

```
SELECT * FROM DBA_PRIV_AUDIT_OPTS;
```

- Ver opciones de la auditoría actual sobre los objetos (*objects*):

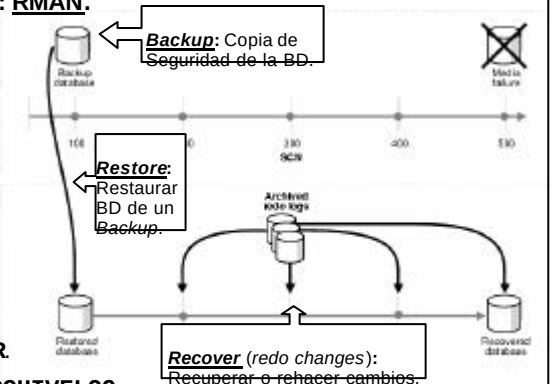
```
SELECT * FROM DBA_OBJ_AUDIT_OPTS
WHERE owner='SCOTT' AND object_name LIKE 'EMP%';
```

Backup y Recovery

- **Fallos en la BD:** Existen muchos elementos que, de fallar, pueden llevar a pérdidas de información de la BD.
  - **Tipos de Fallos:** En el *hardware*, en Oracle, en programas de los usuarios y por errores de los usuarios.
  - **Dificultad para recuperar los datos:** Dependerá de cada caso.
  - El **peor caso** en el que se puede encontrar un administrador es la perdida de uno o varios **discos** en los que se encuentren ***datafiles***.
    - En este caso, el **administrador** no tendrá más remedio que utilizar una **Copia de Seguridad o *BACKUP***, que puede ser total o parcial, en caso de perder total o parcialmente el *tablespace* SYSTEM.
- Más fáciles de resolver son los casos en que no existe perdida de integridad ni en los *datafiles* de la BD ni en los demás ficheros de control.
  - En este caso puede ser suficiente reparar la causa de la avería, rearrancar la máquina y levantar de nuevo las instancias.
  - Oracle llevará a cabo un proceso de recuperación que, caso de terminar correctamente, será capaz de situar a la BD en el momento en que la máquina falló.

Backup y Recovery

- **Tipos de *Backup*:**
  - ***Backup* Lógico:** Copiar los datos de la BD usando el programa Export.
    - Se almacenan en un fichero binario.
  - ***Backup* Físico:** Copiar los ficheros físicos de la BD.
    - Se puede hacer usando las utilidades del **S.O.**, o bien, la herramienta *Oracle8i* ***Recovery Manager*: RMAN.**
    - **RMAN** también permite recuperar datos de un *backup incremental*, que es el que contiene sólo los datos cambiados desde el último *backup*.
    - También se pueden usar los **registros de rehacer archivados** (*redo logs*) usando **RMAN** o las sentencias **RECOVER** or **ALTER DATABASE RECOVER**.



  - La BD debe estar en **modo ARCHIVELOG:**

```
ALTER DATABASE [ARCHIVELOG | NOARCHIVELOG];
```

Backup y Recovery

- **Registros de Transacciones**:
  - Oracle guarda los cambios a la BD en memoria, en el buffer del registro de rehacer (*redo log buffer*): Registros de rehacer (*redo records*).
  - El proceso **LGWR** de Oracle escribe ese buffer en disco frecuentemente en los **Ficheros del Registro de Rehacer Actuales** (***Online Redo Log Files***):
    - Suele hacerse en dos ficheros como mínimo de forma circular: Cuando se llena uno, cambia al otro alternativamente.
    - Si el sistema está en modo **`ARCHIVELOG`**, todos los ficheros del registro de rehacer (*redo logs files*) son copiados y guardados en disco en los llamados **Ficheros del Registro de Rehacer Archivados** (***Archived Redo Log Files***).
    - En modo **`NOARCHIVELOG`** no se guarda ese histórico de cambios y no podrá, por tanto, hacerse una recuperación (*recovery*) total, si falla el sistema.
- **Características de una Buena Estrategia para *Backup-Recovery*:**
  - Tener varias copias de los ficheros de rehacer en distintos discos, y hacer frecuentes *backups*.
  - Guardar los *backups* en sitios seguros diferentes para evitar grandes desastres (incendios, inundaciones...).
  - Un sistema sofisticado es mantener una BD de réplica que se use en caso de cualquier tipo de fallo en la BD original.

Backup y Recovery

- **Estructuras de Datos Importantes para *Backup-Recovery*:**
  - ***Datafiles*** **(ficheros):** Almacenan los datos.
    - En el primer bloque (página) se almacena la cabecera (tamaño, tablespace, fecha de creación, SCN...).
    - Al abrir una BD Oracle compara esa información con la del Fichero de Control para determinar si es necesario un *Recovery*.
  - **Fichero de Control:** Cada BD tiene su fichero de control. Contiene:
    - Nombres en el S.O. de los ficheros de la BD y del registro de rehacer (actuales y archivados), que son leidos al montar la BD,
    - *Checkpoint*: Punto del registro de rehacer,el **mayor SCN**, en el que todos los cambios hechos a la BD antes de ese punto han sido guardados en la BD
      - También contiene el *checkpoint* (mayor SCN) de cada *datafile*.
    - Otros datos importantes: nombre de la BD, fecha de creación, información sobre *backups*...
  - **Segmentos de *Rollback*:** Son estructuras lógicas guardadas en *datafiles* y contienen la información anterior a cada modificación **no confirmada**.
    - El proceso **DBWn** puede escribir los datos en disco aunque no hayan sido confirmados, por lo que los segmentos de rollback son necesarios para garantizar la consistencia a nivel de sentencia y de transacción.
    - Oracle también los necesita para deshacer cambios no confirmados.

Backup y Recovery

- **Ficheros del Registro de Rehacer Actuales (*Online Redo Log Files*):**
  - Guarda todos los cambios hechos a la BD.
  - Al confirmar los cambios, Oracle guarda el SCN.
  - Multiplexados: Si guarda varias copias en diferentes discos, para evitar fallos en estos ficheros.
- **Ficheros del Registro de Rehacer Archivados (*Archived Redo Log Files*):**
  - Son los ficheros **Actuales** ya completos que se archivan en uno o más destinos, si la BD está en modo **`ARCHIVELOG`**.
  - En modo **`ARCHIVELOG`**:
    - Se puede hacer un *Recovery* de la BD con diversas opciones (completo, incompleto...).
    - Se puede hacer un *hot backup* (con la BD abierta y disponible).
    - Se puede mantener una BD de réplica (*standby database*), transmitiéndole y ejecutando estos ficheros.
    - Inconvenientes: Espacio en disco y tareas de gestión de estos ficheros.
  - En modo **`NOARCHIVELOG`**:
    - Sólo se puede hacer un *backup* con la BD cerrada, tras su finalización (*Shutdown*).
    - Casi el único *Recovery* es un *Restore* de la BD, perdiendo todos los cambios hechos desde el último *Backup*.

Backup y Recovery

- **Tipos de *Backup* Físico según su contenido:**
  - ***Backup* de la BD Completa:** Es el *backup* más típico.
  - ***Tablespace Backup*:** Copia de todos los ficheros de cierto *tablespace*.
    - Requiere que la BD esté en modo **`ARCHIVELOG`**, porque al recuperar ese *tablespace* será necesario hacer que sea coherente con el resto de *tablespaces*.
      - **Excepciones:** Si el *tablespace* está en modo de sólo lectura (*read-only*) o desactivado (*offline*). En esos casos, como el *tablespace* no puede modificarse, tampoco hay que rehacer cambios en él.
  - ***Datafile Backup*:** Copia de un *datafile*.
    - Lo normal es necesitar que exista copia también de todos los *datafiles* del *tablespace*, por lo que es mejor usar el tipo anterior.
    - Requiere que la BD esté en modo **`ARCHIVELOG`**.
      - **Excepciones:** Que esté en sólo lectura (*read-only*) o desactivado (*offline*).
  - ***Backup* del Fichero de Control:**
    - Se hace con la BD montada.
    - Sentencia: **`ALTER DATABASE BACKUP CONTROLFILE`**.
    - Oracle permite mantener varias copias del fichero de control (*multiplex*).

Backup y Recovery

- ***Backup* Inconsistente:** *Backup* de uno o más ficheros, hecho mientras la BD está abierta (*open backup*) o cuando se ha caído anormalmente (*inconsistent closed backup*).
  - Es la única opción en BD que deben estar disponibles todo el tiempo.
  - En este caso, no todos los ficheros han sido validados con el mismo SCN:
    - Requiere recuperación (*recovery*) de los datos posiblemente perdidos.
      - Puede usarse **`V$RECOVER_FILE`** para determinar qué ficheros recuperar.
    - Oracle no abrirá la BD hasta que estos SCN sean consistentes, o sea, hasta que todos los cambios de los registros de rehacer actuales (*online redo logs*) se hallan hecho en los *datafiles*.
  - Por tanto, si usamos este tipo de *backup* es buena medida:
    - Forzar a Oracle a archivar todos los registros no archivados
      - **BD abierta :** **`ALTER SYSTEM ARCHIVE LOG CURRENT;`**
      - **BD montada :** **`ALTER SYSTEM ARCHIVE LOG ALL;`**
    - Hacer copia de seguridad del fichero de control y de los registros de rehacer archivados producidos desde que se empezó el *backup*:
      - Esto asegura que el *backup* será útil aunque sea inconsistente y permite borrar esos archivos de rehacer.

Backup y Recovery

- ***Backup* Consistente:** *Backup* de uno o más ficheros, hecho después de cerrar correctamente la BD.
  - Si se cerró mal (por fallo o por usar la sentencia **`SHUTDOWN ABORT`** los ficheros de la BD serán **INCONSISTENTES**, excepto que la BD se abriera en modo de sólo lectura.
  - **Al restaurar una BD (*Restore*):** Oracle determina si el *backup* es o no consistente comparando las cabeceras de los *datafiles* con la información de cada datafile que tiene en el Fichero de Control.
    - Aquí, todos los ficheros han sido validados con el *checkpoint* del mismo **SCN**.
      - **Excepciones:** Si el *tablespace* está en modo de sólo lectura (*read-only*) o desactivado (*offline*) puede tener un SCN más antiguo. Como el *tablespace* no puede modificarse, Oracle no tiene nada que rehacer.
    - La BD puede abrirse **sin recuperar datos** (sin *recovery*).
  - En modo **`NOARCHIVELOG`** esta es la única opción de *backup* válida:
    - Los ficheros pueden ser inconsistentes y para que sean consistentes o cerramos la BD o tenemos la información para rehacer (*redo log*).
    - En este modo, un *backup* inconsistente sólo será útil si los ficheros de rehacer actuales están disponibles cuando se restaure la BD, lo cual es bastante improbable: **RMAN** no permite hacer *backup* de una BD caída anormalmente que corre en modo **`NOARCHIVELOG`**:
      - Si tu BD está en **modo `NOARCHIVELOG`** debes tener una **copia de seguridad consistente** (útil sin hacer *recovery*).

Backup y Recovery

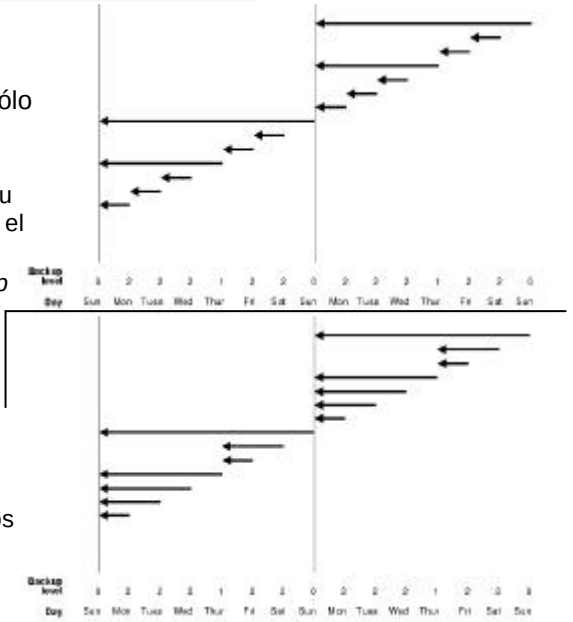
- **Tres Métodos de *Backup*:**
  - **1. RMAN** (*Recovery Manager*)**:** *Backup* físico a partir de Oracle 8.
    - Requiere usar RMAN para operaciones de *Restore* y *Recovery*.
    - El único que admite *backup* incremental.
    - Almacena información sobre el *backup* en el fichero de control.
  - **2. Manualmente, ejecutando comandos del S.O.:** *Backup* físico.
    - Requiere especificar los nombres de los ficheros manualmente.
    - Puede escribirse un *script* para automatizar la operación.
  - **3. Usando la utilidad Export:** *Backup* lógico.
    - Guarda información sobre los objetos del esquema, sin información sobre los *tablespaces*, fichero de control...
    - Requiere utilizar **Import** para recuperar los datos.
- La **Recuperación** (*recovery*) **no** tiene que ser siempre de **toda la BD** y dejar la BD hasta el momento **actual: *Media Recovery*.**
  - Puede interesar hacer una recuperación de una parte sólo y hasta un punto determinado en el tiempo (*incomplete recovery*), porque ciertos datos o una tabla completa se hayan borrado o corrompido...
    - **TSPITR** (*tablespace point-in-time recovery*): Recuperar uno o varios tablespaces a cierto punto en el tiempo.
  - Se puede hacer especificando el punto exacto en el tiempo, el SCN o el número de secuencia en el registro (con RMAN). También es posible efectuar *Media Recovery*, hasta que se cancele manualmente.

1. RMAN: *Recovery Manager*

- **RMAN:** Herramienta para *backup*, copiar, restaurar (*restore*) y recoperar (*recover*) ficheros de datos, de control y registros de rehacer archivados. **Permite, entre otras tareas:**
  - Crear *scripts* con las operaciones frecuentes de *backup*/*recovery*.
  - Crear *backups* incrementales, copiando sólo los datos que hayan cambiado desde el último *backup*.
  - Crear un duplicado de la BD de producción para pruebas.
  - Usar el catálogo (*recovery catalog*) para automatizar tareas de *restore*/*recovery*.
  - Encontrar los *datafiles* que requieren *backup*, usando el límite especificado por el usuario sobre la cantidad de rehacer que debe aplicarse para *recovery*.
  - Comprobar si cierto *backup* puede ser restaurado.
  - Crear informes sobre las operaciones de *backup*.
  - ***Backup* abierto y cerrado:** *Backup* de cualquier parte de la BD cuando está abierto, o cuando la BD está montada pero no abierta. Los *backups* cerrados pueden ser:
    - **Consistentes:** Con la BD montada, no abierta y cerrada correctamente (sin caidas o cierres incorrectos) y que, por tanto, son recuperados sin *recovery*.
    - **Inconsistentes:** Backup de cualquier parte cuando la BD está abierta o cerrada incorrectamente. Requiere *recovery* para que sea consistente.

1. RMAN: *Recovery Manager*

- **Tipos de Backup:**
  - **Total** (*Full*)**:** De todo.
  - **Incremental Multinivel:** Sólo de lo que haya cambiado desde el último *backup*.
    - Cada Incremental tiene su **nivel *n*** (0, 1, 2...), donde el 0 es un *backup* total.
    - A **nivel *n* se** hace *backup* de los cambios hechos desde el *backup* incremental a **nivel *n*, o inferior**.
  - ***Backup* Incremental Acumulativo Multinivel:**
    - El **nivel *n*** de este tipo de *backup*, copia los cambios hechos desde el *backup* más reciente a un **nivel inferior**.



2. *Backup usando el S.O.*

- **Obtener una lista con los Ficheros:**
  - **1. Ficheros que forman la BD** (columna **`name`** de **`V$DATAFILE`**), o cada uno con su *tablespace***,** consultando **`DBA_DATA_FILES`,** o bien**:**

```
SELECT t.name "Tablespace", f.name "Datafile"
  FROM V$TABLESPACE t, V$DATAFILE f
  WHERE t.ts# = f.ts# ORDER BY t.name;
```

  - **2. Ficheros del Registro de Rehacer Actuales** (*online redo log files*)**:**

```
SELECT member FROM V$LOGFILE;
```

  - **3. Ficheros de Control actuales:**

```
SELECT value FROM v$parameter
  WHERE name = 'control_files';
```

- Si el ***Backup*** será **consistente** entonces cerrar la BD. Si será **inconsistente** puede hacerse un *backup* de un único *tablespace*, o de todos:
  - Si el ***tablespace*** no es de sólo lectura, hay que **bloquear** las escrituras:
    - Marcar el comienzo de un *backup online*, estableciendo el estado de sus ficheros como **`INACTIVE`** (columna **`status`** de **`V$BACKUP`**)**:**

```
ALTER TABLESPACE users BEGIN BACKUP;
```

    - Tras hacer el *backup*, desmarcar los tablespaces (estado **`ACTIVE`**)**:**

```
ALTER TABLESPACE users END BACKUP;
```

3. *Backup usando Export/Import*

- Aunque la utilidad real es mover datos, puede usarse para *backup*, pero recupera los datos que se guardaron y se pierden los cambios posteriores.
- **Export:** obtiene una copia consistente de la BD en un momento concreto.
  - Lo ideal es ejecutar la BD en modo restringido, para evitar modificaciones de la BD mientras Export está ejecutándose.
  - **Modos de Export:**
    - ***User*:** Exporta todos los objetos de un usuario.
    - ***Table*:** Exporta todas o algunas de las tablas de un usuario.
    - ***Full Database*:** Exporta la BD completa, con todos sus objetos. **Tipos:**
      - **Export Completo (*Complete*):** Copia toda la BD.
      - **Export Acumulativo (*Cumulative*):** Exporta sólo los datos modificados (y no todo el objeto), desde el último Export *Completo* o *Acumulativo*.
      - **Export Incremental (*Incremental*):** Sólo exporta los objetos modificados desde el último Export (de todo tipo). Exporta definición de objs. y datos.
  - El rol **`EXP_FULL_DATABASE`** permite exportar la BD completa.
  - **Ejemplos:**

```
exp HELP=Y
exp system/manager FULL=Y INCTYPE=COMPLETE
    file=dba.dmp grants=Y rows=Y triggers=N
```


3. *Backup usando Export/Import*

- **Parámetros:**
  - **`HELP=[Y/N]:`** Muestra una ayuda con una descripción de los parámetros.
  - **usuario/contraseña:** Indica la conexión a realizar para realizar el export.
  - **`OWNER=(lista_usuarios):`** Export en modo usuario para dichos usuarios.
  - **`TABLES=(lista_tablas):`** Export en modo Table para exportar dichas tablas.
  - **`FULL=Y:`** Indica que el export será completo.
  - **`INCTYPE={COMPLETE|CUMULATIVE|INCREMENTAL}:`** Tipo de export.
  - **`FILE=Fichero:`** Fichero que la utilidad generará (**`expdat.dmp`** por defecto).
  - **`CONSISTENT=[Y/N]:`** Si Export usa **`SET TRANSACTION READ ONLY`** para asegurar que los datos que Export use sean consistentes. En caso negativo, cada tabla se exporta en una transacción independiente.
  - **`GRANTS=[Y/N]:`** Si queremos que export incluya los permisos de los objetos. Los privilegios del sistema siempre se exportan.
  - **`ROWS=[Y/N]:`** Si queremos que export incluya o no las filas.
  - **`TRIGGERS=[Y/N]:`** Si queremos que export incluya o no los *triggers*.
  - **`PARFILE=Fichero:`** Especifica un fichero con los parámetros de Export.
  - **`INDEXES=[Y/N]:`** Como ya se ha comentado, esta utilidad no asegura la integridad de los datos. Con este parámetro podemos escoger no copiar los índices en el fichero de export y dejar que Oracle los genere otra vez.
  - **`CONSTRAINTS=[Y/N]:`** Si export incluirá las restricciones de las tablas.

3. *Backup usando Export/Import*

- **Import:** Hay que hacer varios Import, según los tipos de Export hechos.
  - El orden en el que se introducen los datos es el siguiente:
    - 1. Se introducen las definiciones de los tipos y de las tablas.
    - 2. Se introducen las filas correspondientes a cada tabla.
    - 3. Se crean los índices y se introduce su información.
    - 4. Finalmente, las restricciones de integridad, las vistas y los *triggers*.
  - Si existían los objetos antes de lanzar el Import podemos tener problemas al utilizarlo: Las restricciones de integridad pueden rechazar la inserción de filas existentes en el fichero del Export.
    - **Soluciones:** Deshabilitar las restricciones y los *triggers*, o hacer que Import se salte alguno de esos pasos.
  - Parámetros **FROMUSER y TOUSER:** Aceptan una lista de usuarios, respectivamente de los que se obtienen los objetos y en los que se recupera la información:
    - **Ej.:** `imp system/manager FROMUSER=miriam TOUSER=joe,patri TABLES=emp`
  - En modo completo (**FULL=Y**) existen dos opciones detectando el Tipo (Completo, Acumulativo o Incremental) **:**
    - **INCTYPE=SYSTEM:** Solo se restaurará los objetos del sistema (como definiciones de objetos...).
    - **INCTYPE=RESTORE:** Se restauran todos los objetos existentes en el fichero tratado sin excepción. Esta es la opción por defecto.
    - **Ej.:** `imp system/manager FULL=Y INCTYPE=RESTORE FILE=Cumulative1`

Herramientas ORACLE

- **Designer 2000:** Facilita el desarrollo
  - **Diagramas Entidad-Relación:** Facilitan la creación de esquemas conceptuales usando esta notación gráfica.
  - **Diagramas Jerárquicos:** Facilita la representación de los procesos de una empresa, usando una técnica de descomposición progresiva de grandes procesos en procesos cada vez más simples y más detallados.
  - **Developer 2000:** Herramienta de programación que permite diseñar interfaces de usuario gráficos (GUI) que requieran transacciones a una BD. Ofrece multitud de herramientas para:
    - Crear Informes, Consultas, Diagramas...
    - Gestionar el desarrollo en equipo.
    - Depurador, que funciona en todos los niveles de la aplicación (Oracle *Procedure Builder*).
    - Permite incorporar controles ActiveX.

Algunas Sentencias SQL del DBA

- El **Comando CREATE** para **Crear** objetos puede sustituirse por **DROP** y **ALTER** para las **Borrar** y **Modificar** el objeto en cuestión**:**
  - **CREATE USER:** Crea un usuario, una cuenta para acceder a la BD.
  - **CREATE ROLE:** Crea un conjunto de privilegios con un nombre.
  - **CREATE SYNONYM:** Crea un sinónimo. Puede establecerse como sinónimo público (sinónimo accesible para todos los usuarios).
  - **CREATE TABLESPACE:** Crea un *tablespace*, espacio en la BD que puede contener objetos.
  - **CREATE ROLLBACK SEGMENT:** Crea un segmento de anulación (*rollback*), un objeto donde Oracle almacena los datos para deshacer modificaciones.
  - **GRANT:** Otorga roles y permisos (o privilegios) del sistema o de objetos a usuarios. Los privilegios se retiran con el comando **REVOKE**.
  - **ANALYZE:** Almacena o borra en el Diccionario de Datos estadísticas sobre el objeto que se especifique. Por ejemplo, para una tabla el resultado se almacenará en **USER_TABLES**.
  - **AUDIT:** Realiza un seguimiento sobre las operaciones ejecutadas o sobre objetos accedidos (usuario, tipo de operación, objeto implicado, fecha y hora). Para detener la auditoria usar **NOAUDIT**. Los datos se guardan en tablas del diccionario con el texto **AUDIT_** en su nombre, como **DBA_AUDIT_OBJECT**, **DBA_AUDIT_TRAIL**...